BOLT BERANEK AND NEWMAN INC

CONSULTING · DEVELOPMENT · RESEARCH

BBN Report No. 3438

SPEECH UNDERSTANDING SYSTEMS

Final Report

November 1974 - October 1976

Volume III: Lexicon, Lexical Retrieval and Control

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Unclassified

(9) Final Technical Progress Rept.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 3438

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

30 Oct 74 - 29 Oct 76

4. TITLE (and Subtitle)

(6) SPEECH UNDERSTANDING SYSTEMS
Final Technical Progress Report
30 October 1974 to 29 October 1976

Volume III. Lexicon, Lexical Retrieval and Control.

5. TYPE OF REPORT & PERIOD COVERED: Final Tech. Prog. Report 30 Oct. 1974 to 29 Oct. 1976

6. PERFORMING ORG. REPORT NUMBER: BBN Report No. 3438

7. AUTHOR(s)

W.Woods, M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue

8. CONTRACT OR GRANT NUMBER(s)

(15) N00014-75-C-0533, ARPA Order-2901

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Mass. 02138

(12) 119 p.

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 5D30

11. CONTROLLING OFFICE NAME AND ADDRESS

ONR
Department of the Navy
Arlington, Virginia 22217

(11)

12. REPORT DATE: December 1976

13. NUMBER OF PAGES: 110

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

(14) BBN-3438-Vol-3

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

(10) William Woods,

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Madeleine Bates,
Geoffrey Brown,

18. SUPPLEMENTARY NOTES

Bertram C. Bruce
Craig C. Cook

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Acoustic-phonetic experiment facility, acoustic-phonetic recognition, acoustic-phonetic rules, artificial intelligence, ATN grammars, audio-response generation, budget management, computational linguistics, control strategies, data base, dictionary expansion, discourse model, fact retrieval, formal command language, formant tracking, grammars, HWIM, knowledge sources,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This five-volume final report is a review of the BBN speech understanding system, covering the last two years of the project from October 1974 through October 1976. The BBN speech understanding project is an effort to develop a continuous speech understanding system which uses syntactic, semantic, and pragmatic support from higher level linguistic knowledge sources to compensate for the inherent acoustic indeterminacies in continuous spoken utterances. These knowledge sources are integrated with

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060100

1/B

19. Key words (cont'd.)

lexical retrieval, likelihood ratio, linear prediction, multi-component systems, natural language retrieval system, natural language understanding, parametric modeling, parsing, pattern recognition, phonetic labeling, phonetic segmentation, phonological rules, phonology, pragmatic grammar, pragmatics, probabilistic labeling, probabilistic lexical retrieval, prosodics, question-answering, recognition strategies, resource allocation, response generation, scoring philosophy, semantic interpretation, semantic networks, semantics, shortfall algorithm, shortfall density, signal processing, spectral matching, speech, speech analysis, speech generation, speech synthesis, speech recognition, speech understanding, SUR, syntax, synthesis-by-rule, system organization, task model, user model, word verification.

20. Abstract (cont'd.)

sophisticated signal processing and acoustic-phonetic analysis of the input signal, to produce a total system for understanding continuous speech. The system contains components for signal analysis, acoustic parameter extraction, acoustic-phonetic analysis of the signal, phonological expansion of the lexicon, lexical matching and retrieval, syntactic analysis and prediction, semantic analysis and prediction, pragmatic evaluation and prediction, and inferential fact retrieval and question answering, as well as synthesized text or spoken output. Those aspects of the system covered in each volume are:

Volume I.	Introduction and Overview
Volume II.	Acoustic Front End
Volume III.	Lexicon, Lexical Retrieval and Control
Volume IV.	Syntax and Semantics
Volume V.	The Travel Budget Manager's Assistant

SPEECH UNDERSTANDING SYSTEMS

Final Report

November 1974 - October 1976

ARPA Order No. 2904

Program Code No. 5D30

Name of Contractor:
Bolt Beranek and Newman Inc.

Effective Date of Contract:
30 October 1974

Contract Expiration Date:
29 October 1976

Amount of Contract: $1,966,927

Contract No. N00014-75-C-0533

Principal Investigator:
William A. Woods
(617) 491-1850 x361

Scientific Officer:
Marvin Denicoff

Title:
SPEECH UNDERSTANDING SYSTEMS

Editor:
Bonnie Nash-Webber
(617) 491-1850 x227

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

The BBN Speech Understanding System: Final Report

This is one of five volumes of the BBN Speech Understanding System Final Report. The Table of Contents for the entire set is given below.

Acknowledgements

To the following people who contributed to the development of the BBN Speech Understanding System:

William A. Woods, Principal Investigator
Madeleine Bates
Geoffrey Brown
Bertram C. Bruce
Laura Gould
Craig C. Cook
Gregory Harris
Dennis H. Klatt
John W. Klovstad
John I. Makhoul
Bonnie L. Nash-Webber
Richard M. Schwartz
Jared J. Wolf
Victor W. Zue

To our secretaries - Beverly Tobiason, Kathleen Starr and Angela Beckwith - for the exceptional diligence, competence, and good humor they have shown throughout the assembly of this report.

"Had we but world enough and time..."
Andrew Marvell, To His Coy Mistress

TABLE OF CONTENTS

A. DICTIONARY

Two dictionaries exist for HWIM's travel budget management domain, differing primarily in their number of lexical items. TRAVELDICT, having 491 lexical items and compatible with both SMALLGRAM and MIDGRAM, (Vol. IV, Sec. B) has been used extensively throughout the development of our system. BIGDICT, having 1,138 lexical items and compatible with BIGGRAM, fulfills the 1000-word-vocabulary requirement set forth in the Newell report [Newell et al., 1973]. This larger dictionary, however, was not used until the final performance test of our complete system.

Our major considerations in constructing the dictionaries were to provide the user with a habitable and reasonably complete way of discussing the domain and to provide our acoustic-phonetic front end with a wide range of phonetic and phonological situations. Thus, several methods were used in setting them up.

(1) A set of domain-related sentences was collected, and all words in these sentences and in the sentences from incremental simulations were included in the dictionary;

(2) Roget's Thesaurus was used to find useful synonyms of existing words, to increase the number of ways things could be said;

(3) The first and last names of all BBN Division 4 personnel were included,

(4) As were the names of all Division 4 projects and their sponsors;

(5) Many place names (cities, states, countries, universities, etc.) were added. All places appearing as the destinations of trips recorded in the data base are represented in the dictionary; further place names were obtained from meetings announced in the ACM calendar.

Among the types of lexical entries in these dictionaries are the root form of regularly inflected words, the root and inflected forms of irregularly inflected words, and a small number of compound words like "New-York," and "round-trip." For each item, the dictionaries contain its pronunciation baseform, as given in Kenyon and Knott's A Pronouncing Dictionary of American English, syntactic-semantic class and feature information, and, where appropriate, manner of inflection. The class information corresponds to the categories recognized on CAT arcs of the grammar. Thus, beside such standard classes as noun, verb, adjective, etc., the DATE/ and proper NAME/ networks of the grammar recognize such

classes as MONTH, WEEKDAY, FIRSTNAME, LASTNAME, and CITY. The lexical items are well distributed over the classes, giving the user of the system a comfortable subset of English words in which to express his thoughts.

The pronunciation base forms of the lexical entries are represented in the 48 phonemes of the modified ARPAbet, and four levels of stress are distinguished. The stress indicators are:

!- reduced stress (following front, back and retroflexed schwas, syllabic nasals and syllabic [L].)

!0 unstressed

!1 auxiliary stress

!2 primary stress

Syllable boundaries are marked in the dictionary with the symbol *. This information is accessed during dictionary expansion by phonological rules such as R16.1, which syllabifies R when it occurs after a syllable boundary, as in "arrange".

```
[R16.1
        (RULE ((* AX !- * R)
               (* AXR !- * 0)) ]
```

(For a complete list of the phonological rules used by HWIM, see Appendix 1.) Inflectional information for nouns, verbs and adjectives is given in terms of orthographic spelling, and the inflected forms are then generated by rule during the first phase of dictionary expansion (Sec. C). For example, the lexical item "figure" has the property V S-D, indicating that as a verb, "figure" forms its third singular present form orthographically by adding an "s" and its past and past participle forms by adding a "d".

Complete listings of TRAVELDICT and BIGDICT can be found in Appendix 5 of this report.

B. PHONOLOGICAL RULES

A necessary consideration in the design of a speech understanding system is the incorporation of various sources of phonological knowledge [Oshika et al., 1975]. It is well known that words can be pronounced in continuous speech in a number of different ways (depending on the speaker as well as the linguistic environment), resulting in acoustic signals that are sometimes quite different. A speech understanding system intended to handle a large lexicon and multiple speakers will undoubtedly have to capture such phonological variations, and mediate between the lexical entries and their acoustic realizations.

It is also clear that the sound of a word will vary with its phonological environment, such as the surrounding vowels or consonents or the stress patterns, and that many of the sytematic relationships can be captured in general phonological rules. The development of such rules, therefore constitutes a necessary and intgral part of speech understanding research.

Phonological variations in continuous speech are being dealt with in HWIM at two different levels. Within-word phonological effects are accounted for in a dictionary expansion phase that generates for each word in the lexicon all its possible stand-alone pronunciations. Across-word phonological effects are later taken into account in the dictionary compilation procedure of the Lexical Retrieval component [Sec. D].

1. Within-word Phonological Effects

Our approach to solving the problem of within-word phonological effects can be characterized as follows: for each word in the dictionary, a minimal and appropriate set of lexical base forms are assumed. This set of baseforms is then augmented via a collection of generative phonological rules into a complete set of pronunciations for the given word [Woods and Zue, 1976].

Lexical base forms are derived from the pronunciations found in Kenyon and Knott's A Pronouncing Dictionary of American English, appropriately modified to reflect the performance of our Acoustic-Phonetic Recognition program. For example, since our APR program distinguishes between front

and back schwas, whereas A Pronouncing Dictionary of American English does not, our base forms contain both front and back schwas.

In the process of dictionary expansion, a set of phonological rules is used that covers a broad spectrum of variations ranging from syntactic inflections (plural, tense, etc.) to vowel reduction, consonant syllabification and palatalization.

It should be noted that the generative phonological rules in the system account for both general phonological effects and ones that are dependent on the performance of the APR. Rules in the latter category are of the two kinds. Rules of the first kind reflect the type of allophonic variations distinguished by APR and are applied obligatorily. For example, [K's] are separated into front and back [K's], depending on the features of the following segment. Rules of the second kind reflect genuine shortcomings of the APR and are applied optionally. For example, since the APR cannot detect a voiceless schwa following a voiceless aspirated plosive, the schwa is simply deleted by rule.

An annotated list of rules used for dictionary expansion is included in Appendix 1. It should be noted that this list represents a subset of all the phonological rules that we have collected and developed during the course of our research. The rules that we have chosen for dictionary expansion reflect either phenomena that we have seen in our acoustic data or phenomena so generally common that we can expect their occurrence. Some of the more exotic rules, such as [r]-deletion and vowel raising, have been excluded from the list so that the size of the expanded dictionary will remain manageable.

2. Across-word Phonological Effects

Across-word phonological effects are described by an unordered set of optional rules, which are applied to the complete set of word pronunciations that result from the consideration of within-word phonological effects. In effect, they are applied to all word pairs in the dictionary that satisfy the contextual constraints. (See Sec. D for a fuller discussion of across-word phonological rules.)

Appendix 1 also includes an annotated listing of a program, written in BCPL, used to generate all across-word phonological rules. It should be noted that although there exists a large amount of overlap between the two rule sets, there are certain rules that only appear in the set describing across-word phonological effects. For example, rules are provided to insert a glottal stop between two vowels across a word boundary, and unreleased stops are allowed at word- and phrase-final positions.

C. DICTIONARY EXPANSIONS

1. Historical Perspective

In the first two years of the speech project (1971-73), we developed two different kinds of phonological rule mechanisms whose effectiveness we were able to compare. The first was a capability to inspect the segment lattice of phonetic labelings and apply phonological rules in reverse to add additional branches to the lattice. The second was a generative mechanism to apply phonological rules in the forward direction to the pronunciations of the words in the lexicon to produce additional possible pronunciations (i.e., dictionary expansion). We envisioned that certain phonological rules would be best implemented in one direction and others in the other direction. In the current system (with the exception of some of the rules used in the APR component itself), we are using only forward generative rules. The reason for this shift to exclusively generative rules is a combination of the disadvantages of inverse rules and the discovery of an efficient solution to our major problem in the use of generative rules.

Applying phonological rules in reverse to the segment lattice has two principal disadvantages. First, there is no good way to implement obligatory rules in the reverse direction. There is in general no easy way to remove from the segment lattice just that path that is matched by a rule without also killing other paths that are not matched by the rule. Even if there were such a method, a path matched by a rule might arise accidentally and not by the mechanism corresponding to the rule, and therefore, in the reverse direction most rules must be considered optional and not obligatory. As a result, the effect of inverse rules is almost always to increase the subsequent processing required. Secondly, the identification of the segments in the lattice is only tentative and may be in error. Hence, many rule matches are also likely to be in error. Looking at such a lattice, one finds possible sequences of labelings that one would not expect to find in idealized pronunciations, and consequently the rules must be applied in all sorts of cases that they would not have faced in the generative direction. Moreover, when the segment lattice includes a score

for every possible phoneme at every point, as it currently does, many rules match very low scoring sequences, and it is difficult to decide how to assign scores to the additional branches added to the lattice.

In the generative direction, on the other hand, it first appeare hat the across-word phonological rules could not be applied to a word until one knew its context. Since the set of possible sentences that could be constructed is potentially infinite, the complete string of phonemes to which one would like to apply rules does not exist until a complete sentence has been hypothesized. Yet, one may need to have applied rules to a word in order to know whether it has occurred in the input in the first place. The problem appeared to be a vicious circle. However, the circle can be broken by a technique developed by Klovstad [Klovstad and Mondschein, 1975; Klovstad, 1976], which permits across-word-boundary phonological rules to be applied when the in-core dictionary is being constructed. With the resulting compiled dictionary structure, all word matches (including those with word-boundary effects) can be found at run time and the contextual constraints implied by the use of a rule are automatically propagated to adjacent words. This technique is described more fully in [Klovstad, 1976].

2. Phases of Expansion

The phonological rules used in dictionary expansion are divided into several different groups and applied at several different phases in the process. The within-word rules are divided into three classes. First, there is a group of rules that produce the pronunciations of regular inflections. These rules are used in an initial phase of dictionary expansion that augments the dictionary with the regularly inflected forms of verbs, adjectives, and nouns. They handle contextual dependencies such as the pronunciation of plural endings of nouns depending on whether the preceding phoneme is voiced, strident, or neither.

A second set of general English phonological rules applies to the output of the inflectional rules to produce alternative pronunciations and obligatory transformations of the base pronunciations. These rules correspond closely to the rules that are recognized, named, and studied by

phonologists. They include such rules as palatalization, syllabification, vowel reduction, glottalization, etc., details of which are given in Section B and in the appendices to this volume.

A third set of within-word rules handles phonological phenomena that are consequences of the particular capabilities of the acoustic-phonetic recognition component of the system. These APR rules introduce context-specific allophones of several classes of phoneme in places where the APR is capable of making more precise discriminations, and they compensate for such things as the inability of the APR to detect nasalization in vowels. The APR rules thus adjust the word pronunciations to make them more specific in some places and less specific in other places, to match the discrimination abilities of the acoustic-phonetic front end.

After the within-word rules have been used for dictionary expansion, a separate set of across-word boundary rules are applied when the in-core dictionary tree for the Lexical Retrieval component is being constructed [Klovstad, 1976]. Frequently, the same rule will apply both within a word and across word boundaries. In this case, the rule will be written twice, once in the format and conventions for across-word-boundary rules, and once in the format for within-word rules. Table 1 shows the growth of BIGDICT through dictionary expansion. (The parenthesized figure, 1097, refers to the number of words actually recognized by the BIGGRAM grammar.)

	BIGDICT	Incremental Expansion Ratio
Roots & irregularly inflected forms	1138	
Roots & all inflected forms (words)	1363 (1097)	1.20
Pronunciation baseforms	1789	1.31
Pronunciations following first application of phonological rules	3308	1.85
Pronunciations following second application of acoustic-phonetic rules	3771	1.14
Pronunciations following application of across-word phonological rules	8642	2.29

Table 1.

3. Rule Expansion System

All of the within-word phonological rules are written in a computer readable variant of the notation that phonologists generally use for publication (e.g., [Chomsky & Halle, 1968]). The program that uses these rules is an extension of a program written by Bobrow and Fraser [1968] for interactive testing of phonological rules. That program provided for the definition of phonemes, rules, and word pronunciations and for the expansion of a given word by a given list of rules. It did not provide for the automatic generation of all possible pronunciations from a given set of optional rules, nor did it provide a good way to systematically expand all of the words in a vocabulary and produce a tabulation of results.

The extensions to the Bobrow-Fraser system that we have implemented provide for the specification of optional rules with likelihood estimates for the alternatives of applying and not applying them. They provide for the conditional testing of rules depending on the success or failure of previous rule matches, the presence of various pronunciation features associated with a pronunciation, and arbitrary predicates that can be written in LISP to test any desired property of a word. The extensions also provide for the addition of pronunciation features to a word conditional on the success of previous rules.

Since the rule expansion system has already been described thoroughly [Woods, 1975; Woods and Zue, 1976], we will not do so here, except to mention subsequent changes and detail the terminal instructions required to construct the expanded dictionaries.

4. Products of the Expansion

The different phases of dictionary expansion result in several different products, which are used in various places in the speech understanding system. The initial expansion for inflectional rule application is done primarily to produce the input to subsequent phases and to produce a VERDICT file which is used by the Verification component. The VERDICT file contains appropriate base-form pronunciations for all words in

the vocabulary -- both inflected and uninflected -- but does not have all of the different pronunciations that the phonological rule expansion will later introduce. Many of these phonological effects are dealt with separately in the synthesis-by-rule section of the Verification component, and others (principally those such as syllabification, which involve ambiguity of segmentation into distinct phonemes) are handled automatically by the parametric matching procedure.

After the initial inflectional-ending rules are applied, the main phonological rule expansion phase occurs to produce the various pronunciations that the system is to consider for each word. The result of this expansion is an EXPDICT file, which contains not only phonological information, but also syntactic information used by the Control and Syntactic components of the system. There are places in the grammar (such as checking agreement of the determiners "a" and "an") where the pronunciations of the words are important as well as the syntactic categories and semantic markers. The Control component, on the other hand, occasionally wants to know whether a given word could start or end a sentence and may check syntactic category information to determine this.

The final APR-rule expansion is done to produce the set of pronunciations (the DICTTEXT file) that the Lexical Retrieval component will use in constructing its in-core dictionary tree.

Various other files are constructed as side effects of dictionary expansion. Among these are EXPHONES and APRPHONES files, which provide a complete synopsis for the phonologist of all the rules that were used and which words they applied to, as well as the list of the rules that successfully matched each word. Summary statistics, such as the number of pronunciations before and after expansion, are also computed. The formats and examples of the various dictionary files produced are given in Appendix 2.

5. Backup Protection During Expansion

A major new feature of the EXPFILE function that has recently been incorporated is an automatic backup capability to enable the user to

resume a dictionary expansion from a backup sysout in the event of a system crash. This feature was important in expanding the 1000-word dictionary since the amount of time that could have been lost by a system crash was considerable. The backup feature provides for the construction of a sysout file named SAVED.EXPFILE after the expansion of each 100 words. The system automatically deletes all but the latest two versions of this file each time it makes a new one, and when the expansion is complete, it deletes the remaining ones. If the computer should crash during the run, the latest version of this file can later be used to continue the expansion from the point of the backup. The SAVED.EXPFILE sysout remembers the positions of the file pointers and the status of all open files at the time of the backup and automatically restores this state and continues the expansion when it is run.

Appendix 6 gives the necessary details for using the dictionary expansion program.

D. LEXICAL RETRIEVAL

Introduction

This section discusses some aspects of HWIM's Lexical Retrieval component that have not been previously dealt with in depth. (See [Klovstad, 1976] for a detailed description of the material taken for granted here.) The Lexical Retrieval component is a highly refined program for determining the n most probable word matches in a full lexicon or some subset of it appropriately defined by a list of possible words and categories. It operates on a phonetic segment lattice whose segments are described probabilistically (Vol. II, Sec. B). The word matches that it returns are ordered by probability score and specify such information as location and scan direction. The Lexical Retrieval program makes use of a distributed key representation of the dictionary that merges common parts of different words. This makes an effective search of the entire dictionary computationally feasible without having to consider each word separately. The program provides for matching words both left-to-right and right-to-left, taking appropriate account of across-word-boundary phonological effects [Klovstad, 1976]. In actual sentence recognition, the efficiency of this component makes it reasonable for HWIM's Syntactic component to predict all possible words and categories that are acceptable adjacent to a given island.

We begin with a brief review of our scoring philosophy, followed by a detailed description of its implementation at the acoustic-phonetic level. We then discuss how Lexical Retrieval's matching strategies evolved within the context of HWIM and how it is currently being employed there. Finally, we describe some tests that were undertaken to evaluate its stand-alone performance.

1. Implementation of Scoring Philosophy

A scoring philosophy [Klovstad, 1976] specifies how judgments due to many different knowledge sources can be integrated, each providing its own evaluation of a given hypothesis in the form of a probability. These probabilities can then be multiplied together to produce a single one which

can be used as a global measure of quality. The evaluation due to acoustic-phonetic knowledge is of particular interest for at least two reasons:

a) It has the biggest effect on the global probability measure. This can be appreciated by noticing both the frequency and magnitude of its contributions as specified by the scoring philosophy.

b) It requires determining the probability of a discrete sequence (the phonemes) given a continuous waveform (the acoustics).

This evaluation is discussed extensively here because of its unique implementation considerations. That is, any implementation requires both:

a) establishing, for each phoneme in a sequence to be scored, a correspondence with some segment of the acoustic waveform;

b) calculating the probablity of that phoneme sequence given the waveform in the corresponding acoustic segments.

These two requirements parallel the two major operations of the APR: a) partitioning the acoustic waveform into consecutive temporal segments and b) labeling each of them based on the observed acoustics in that segment. As these operations have already been discussed in detail (Vol. II, Sec. B), we will focus our discussion on the segmental level.

a) Review of the Scoring Philosophy

At the acoustic-phonetic level, our scoring philosophy assigns a score S_i to a phonetic sequence P_i in proportion to its probability given the acoustic input A (i.e., $P(P_i|A)$). In order to compute $P(P_i|A)$, we rewrite it as $P(P_i)*P(A|P_i)/P(A)$ using Bayes rule Each word in our lexicon has one or more pronunciations, P_i, each with an estimated a priori probability, $P(P_i)$, that pronunciation P_i will be used for this word. (These a priori probabilities are computed by rule during dictionary expansion, as described in Section C.) To rank a word with pronunciation P_i, we currently calculate $P(A|P_i)/P(A)$ and report both it and $P(P_i)$ to Control as a separate scores. (These probabilities are kept separate so that their usefulness to the system can be investigated at the discretion of Control.) $P(A|P_i)|P(A)$ is commonly referred to as the likelihood ratio. (In this context the words "pronunciation" and "phonetic sequence" are used interchangeably.) Note that in comparing two pronunciations P_i and P_j that span the same acoustics A, ranking will depend only on $P(A|P_i)$ and $P(A|P_j)$.

Most often, however, pronunciations span different parts of the acoustic input, so likelihood ratios are calculated.

In our application, P_i is rarely a single phoneme but rather a phoneme sequence, thus posing the question of how $P(A|P_i)$ should be evaluated. The Lexical Retrieval component does not work off the raw acoustics of an utterance but rather a lattice representation of it segmented into labeled regions. Each such segment has a probabilistic descriptor, which contains for every phoneme an estimate of the probability of its underlying the region. An evaluation of $P(A|P_i)$ then requires establishing a linear correspondence between the phonemes in P_i and the segments associated with the acoustics A.

b) Segmentation Problems

Unfortunately, the APR occasionally makes segmentation errors, so we cannot count on a one-to-one alignment of each phoneme in the pronunciation with a segment in the segment lattice. The most frequently observed segmentation errors are due to the APR's 'missing' a real segment boundary or 'finding' a non-existent one. The first has the effect of producing one segment where there ought to have been two. The second error produces two segments where there ough to have been one. Actually these merely account for the first order effects: 'missing' two or more consecutive real boundaries or 'finding' two or more consecutive non-existent boundaries are also possible, though very unlikely.

c) Path Alignment

In this section we describe path alignment (the alignment of a single pronunciation against segments in the lattice) in terms of incremental alignments. Three kinds of incremental alignments - "match", "merge" and "split" - are involved. A match refers to the incremental alignment of a single phoneme with a single segment. A merge refers to the alignment of a single phoneme with two adjacent segments. A split refers to the alignment of two adjacent phonemes with a single segment. Any path alignment that can be generated by a unique sequence of incremental alignments is possible. The only constraint on the generation of path alignments from

incremental alignments is that each phoneme in the path and segment in the lattice may participate in only a single incremental alignment.

Although many other kinds of incremental alignments are conceivable (corresponding to the higher order segmentation errors mentioned previously), these three alignments permit sufficient flexibility to compensate for most segmentation errors, since higher order errors are very infrequent. (Of the total number of incremental alignments required to compensate for the current APR segmentation behavior, 96.35 percent are matches, 1.7 percent are merge and 1.6 percent are matches. The remaining .35 percent corresponds to higher order errors.) When they do occur, a "correct" alignment is impossible, although a good scoring one may often be obtained. For this reason, the extra computational effort required to correctly compensate for higher order segmentation errors does not appear to be justified, and the possibility of their occurrence is not taken into account.

Figure 1 illustrates in terms of incremental alignments all possible path alignments up to length 4. In this figure, incremental alignments are represented as "I", "/\", and "\/" for match, merge, and split, respectively, and path alignments are pictured as nodes in the tree. Alignments of length n can be generated incrementally by concatenating one additional incremental alignment to its predecessor's alignment of length n-1. The predecessor's alignment may be altered if the last phoneme was aligned in a match, in which case the last phoneme may be realigned as a merge.

If there were no uncertainty in producing the phoneme to segment correspondence, the probability of a given phoneme sequence could be calculated directly from the APR scores. As we have just noted, however, there is some uncertainty as to whether the segmentation produced by the APR is in fact correct. By correct, we mean that there exists some path through the lattice that permits a one-to-one correspondence between phonemes and acoustic segments, and that the boundaries between adjacent segments really 'separate' the acoustics of the two corresponding adjacent phonemes. Our probabilistic scoring philosophy permits handling this

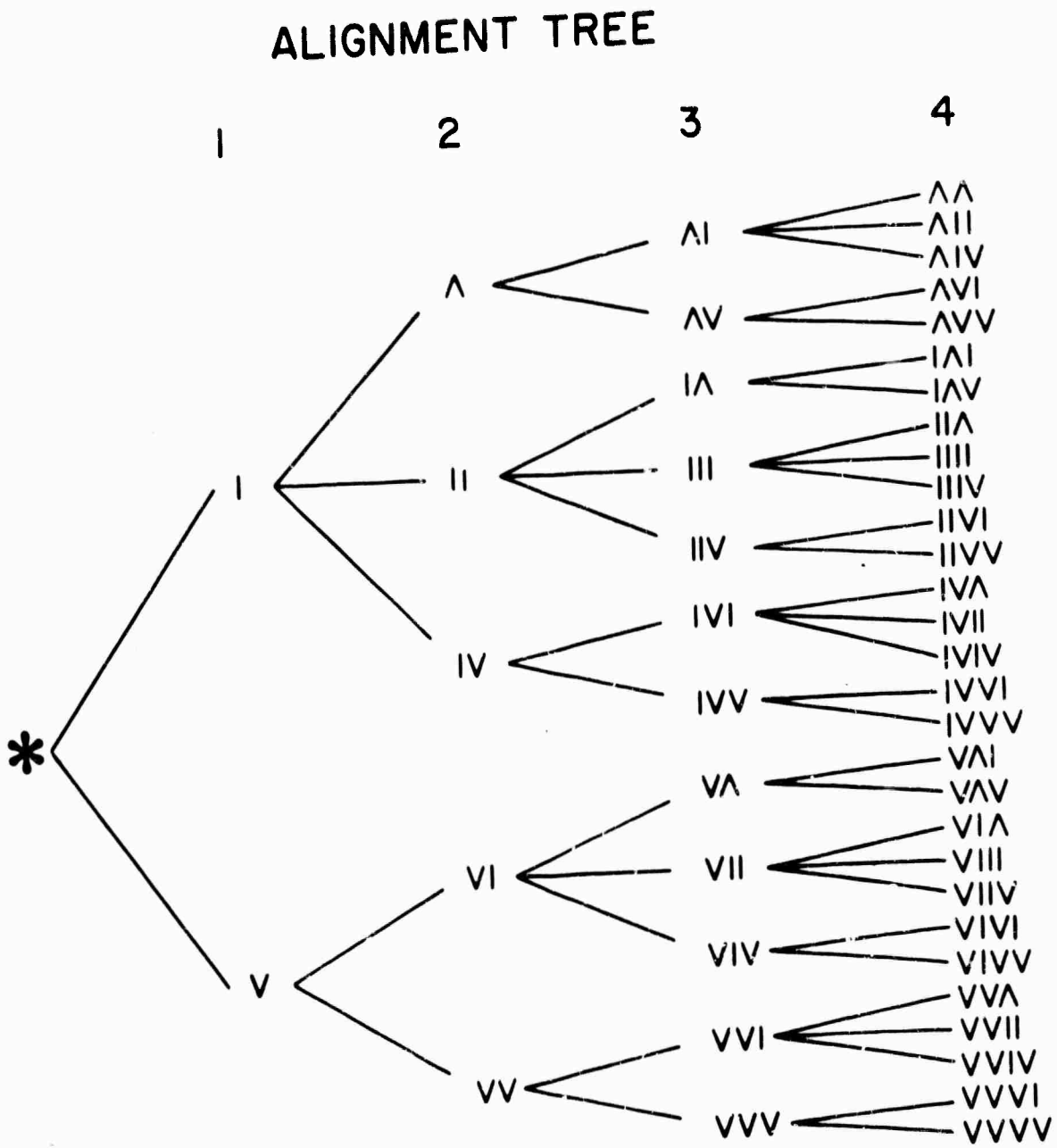

ALIGNMENT TREE
1
2
3
4
*
I
V
Λ
II
IV
VI
VV
ΛΙ
ΛV
ΙΛ
ΙΙΙ
ΙΙV
ΙVΙ
ΙVV
VΛ
VΙΙ
VIV
VVΙ
VVV
ΛΛ
ΛΙΙ
ΛIV
ΛVΙ
ΛVV
ΙΛΙ
ΙΛV
ΙΙΛ
ΙΙΙΙ
ΙΙΙV
ΙΙVΙ
ΙΙVV
ΙVΛ
ΙVΙΙ
ΙVIV
ΙVVΙ
ΙVVV
VΛΙ
VΛV
VΙΛ
VΙΙΙ
VΙΙV
VIVΙ
VIVV
VVΛ
VVΙΙ
VVIV
VVVΙ
VVVV

Figure 1.

uncertainty by considering all possible path alignments. Specifically, the probability of a phoneme sequence given an uncertain segmentation is defined as a weighted sum of the probabilities given each specific path alignment. The weight in each of these terms is the probability of that particular path alignment.

To make things explicit, let us consider how $P(A|P_i)$ would be computed given the phoneme sequence P_i = (P1 P2 P3 P4) and the segmented acoustic waveform A = (S1 S2 S3 S4). Substituting for A and P_i we have:

$P(A|P_i)$ = P(S1 S2 S3 S4|P1 P2 P3 P4)

From Figure 1 we see that the only possible path alignments are those shown in Figure 2.

By summing over all these path alignments we find that:

$$P(S1\ S2\ S3\ S4|P1\ P2\ P3\ P4) = \sum_{i=1}^{7} P(S1\ S2\ S3\ S4, \text{alignment } i|P1\ P2\ P3\ P4)$$

$$= \sum_{i=1}^{7} \underbrace{P(\text{alignment } i|P1\ P2\ P3\ P4)}_{\text{alignment weight}} * \underbrace{P(S1\ S2\ S3\ S4|P1\ P2\ P3\ P4, \text{alignment } i)}_{\text{acoustic phonetic probability conditioned on alignment } i}$$

In our current implementation we make a simplifying approximation, choosing the largest of these terms as the alignment score. Since there are very few segmentation errors (Vol. II, Sec. B), this is generally a very good approximation. The worst approximation occurs when there are severe segmentation errors present, since in this case it is possible that two or more of the path alignments may produce scores of the same magnitude.

d) Path Scoring

The scoring philosophy indicates that the score for a single path alignment is the product of the incremental alignment scores. The incremental score due to a match can be closely approximated, since it is provided directly by the APR as part of the segment description, but accurate score approximation for split and merge alignments is more difficult and is handled exclusively within the Lexical Retrieval

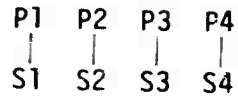
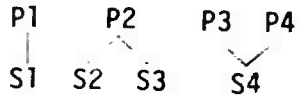
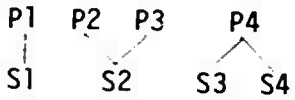
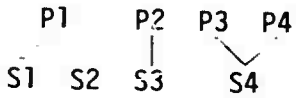
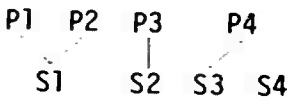
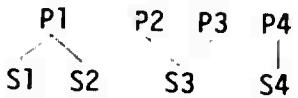
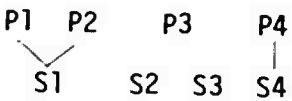
Alignment Number	Alignments	Weighted Alignment Probability
1		P(S1,match|P1)*P(S2,match|P2)*P(S3,match|P3)*P(S4,match|P4)
2		P(S1,match|P1)*P(S2 S3,merge|P2)*P(S4,split|P3 P4)
3		P(S1,match|P1)*P(S2,split|P2 P3)*P(S3 S4,merge|P4)
4		P(S1 S2,merge|P1)*P(S3,match|P2)*P(S4,split|P3 P4)
5		P(S1,split|P1 P2)*P(S2,match|P3)*P(S3 S4,merge|P4)
6		P(S1 S2,merge|P1)*P(S3,split|P2 P3)*P(S4,match|P4)
7		P(S1,split|P1 P2)*P(S2 S3,merge|P3)*P(S4,match|P4)

Figure 2.

component. In order to estimate these probabilities, we have created two scoring matrices, one for splits and one for merges, based on labeled split and merge alignments found in our data base. Because of the infrequent occurrence of these kinds of segmentation errors and the rather modest proportions of our data base, the possibility of accurately estimating each of the nearly million split or merge combinations (915,975 for splits and 723,345 for merges) was clearly out of the question. Therefore, we made an approximation that resulted in a much more manageable problem. Our solution was to map the segments (labels) and phonemes into segment and phoneme classes respectively, thereby permitting the creation of substantially smaller split and merge matrices indexed on these classes.

2. Evolution of Matching Strategies

In matching pronunciations against the segmented acoustic input, the need to accommodate context-sensitive word boundary effects, especially when context has not yet been determined, imposes stringent constraints on effective dictionary structures [Klovstad, 1976] as well as on the matching algorithm discussed above.

This section describes and compares three strategies for handling word boundary effects that were tested during the development of the lexical Retrieval component at BBN. The two special dictionary structures necessary have already been described in considerable detail [Klovstad, 1976]. The first of these structures was designed to permit word recognition based on the kernel of the pronunciation. This is the part of the pronunciation that is unaffected by word boundary rules. For convenience we will refer to this as the Pre-wbe structure (pre-word boundary effect). The second structure was designed to permit word recognition based on the kernel plus some portion of the word boundary effect. (The amount of observed word boundary effect can be specified explicitly in each word boundary rule.) This will be called the Post-wbe structure. In correspondence with these two structures, the three following strategies were tried and tested.

a) Pre-wbe strategy

Scans are made using the Pre-wbe structure. Finding phonologically consistent adjacent words involves resuming the scan in the Pre-wbe structure where the previous scan had terminated.

b) Post-wbe strategy

Scans are made using the Post-wbe structure. Finding phonologically consistent adjacent words involves resuming the scan in the Post-wbe structure where the previous scan had terminated.

c) Post-wbe strategy with Pre-wbe Backtrack

Scans are made using the Post-wbe structure. During the scan, when the Pre-wbe point is passed, information about the current incremental alignment and lattice position is remembered. Finding phonologically consistent adjacent words involves backtracking into the Post-wbe structure to the Pre-wbe position, and resuming from there.

These strategies were tried and tested in the order stated above, each being an attempt to lessen the problems observed in its predecessor.

a) Pre-wbe strategy

The kernel (K) of a pronunciation is defined with respect to a given set of phonological word boundary rules. If P is a phonetic spelling and R is the set of word boundary rules whose left context is satisfied by P, then the kernel K is defined as the longest left substring of P that would be unchanged by the application of any of these rules R.

A Kernel tree is formed by merging common initial phoneme sequences of word pronunciation kernels such that each path in the tree specifies a unique phonetic sequence. As a result, no two distinct paths through the tree are associated with the same phonetic sequence, and no two distinct initial phonetic sequences are associated with the same path. A pronunciation path is a path that terminates on a node in the tree at which some word is identified.

After the Kernel tree has been completed, additional paths must be added to permit the transformations specified in the rules and to complete the kernels (if necessary). For every pronunciation, an entry tree is constructed containing just these necessary paths. One entry tree can be shared by different pronunciations that satisfy the same set of word boundary rules.

Each entry tree contains one path for the kernel residue. This is the portion of the pronunciation that can be changed by a word boundary rule. (The original pronunciation can be generated by concatenating the kernel and the kernel residue.) At the node terminating this path, there is a

branch that enters the root of the Kernel tree. The other paths provide for the transformations in the rule subset. At each node that completes a transformation, there is a branch that enters the Kernel tree at a node determined by the right context of the corresponding rule. The set of words that satisfy its right context are precisely those words that start with the specified phonetic sequence. The set of words is then uniquely specified by the node in the tree that terminates the right context.

Now by concatenating these pronunciations - continuing each kernel by following paths in the corresponding entry tree - one observes that all sequences of pronunciations and possible word boundary effects can be generated. A more detailed description with examples is available in [Klovstad, 1976].

These kernel and entry trees constitute the necessary structures for the Pre-wbe strategy. In this strategy, scans for word matches phonologically compatible with a given word match are begun using the specially constructed entry tree associated with each pronunciation in the kernel tree. In such a scan, the path necessary for kernel completion is considered along with all and only those paths that correspond to compatible word boundary rules. All scans, whether started at the root of the Kernel tree as an initial scan or at the root of a particular entry tree as an anchored scan, stop matching a word when its kernel is completed (i.e., the Pre-wbe point in one of its pronunciations is reached). Continued scans from this point (that is, the root of the kernel's entry tree) for compatible matches are possible.

There are two main advantages to the Pre-wbe strategy:

a) The Pre-wbe structure is less complex and requires less memory than the Post-wbe structure

b) It is a less complex and more efficient search strategy than the others.

However, its performance is deficient with respect to the following two problems:

a) A correct but poorly scoring word would be eliminated when this could have been prevented by the high score of some robust phoneme outside its kernel. (i.e., beyond the Pre-wbe stopping point)

b) An incorrect word would outscore correct words when none of the wbe or kernel completion paths would have scored well and would have lowered the overall score.

These problems are generally more severe for short words, since the unscored phonemes are more likely to be a significant portion of the total word. Both problems were observed occasionally and were at least partially responsible for the unsuccessful recognition of some sentences.

b) Post-wbe strategy

It was believed that scoring phonemes beyond the end of the kernel would eliminate both of the above problems, and the Kernel tree with embedded word boundary rules was used as a guide in the consideration of an alternative structure.

In this new structure the distinction between Kernel and entry tree is blurred somewhat, for the transition from one to the other is now recognizable only by special marks. Now if a specially marked node is reached while scoring in the Kernel tree, information that identifies the particular word and pronunciation being scored is picked up and saved, and scoring continues on into the appropriate entry tree. At some node in the entry tree, a second and different mark indicates that scoring had proceeded far enough. Kernel paths that represent complete pronunciations end at unmarked nodes in the Kernel tree. With this strategy two word matches were said to be equivalent only if they spanned identical portions of the segment lattice (i.e., their starting and stopping segments are identical), they both reference the same word (i.e., the same orthographic spelling), and they both continue on from the same node in the transition tree (same word boundary effects). Our experience with this strategy confirmed its anticipated defects:

a) The list of words returned on each scan tends to contain many variants of essentially the same word match, the only difference being possible word boundary effects.

b) The left and right context specifications in the word boundary rules are not accurate enough, being either too narrow or too broad. In the former case, the boundary effect on an otherwise correct word match would be incompatible with the word that actually adjoined it. In the latter, an incorrect word would be permitted adjacent to a correct word by virtue of too lax a specification.

c) Post-wbe strategy with Pre-wbe Backtrack

Although the Post-wbe strategy eliminated the deficiencies noted for the Pre-wbe strategy, it introduced many new word match variants, thereby reducing system efficiency. To eliminate some of these word matches variants, we changed our concept of word match equivalence. Remembering that there were fewer word match variants when the scoring stopped at the kernel, we decided to define equivalence with respect to kernel word matches as we had in the Pre-wbe strategy. We continue to score beyond the kernel as in the Post-wbe strategy, but we pick up and save (in addition to word and pronunciation information) both score and boundary information when we reach the specially marked node indicating the end of the kernel. We now use: a) the kernel boundary in place of the stopping segment for determining spanning equivalence, and b) the kernel's entry tree in place of the stopping node in the transition tree for determining word boundary rule equivalence. Since we now save the kernel boundary, we are able to start each subsequent adjacent word scan at the root of the kernel's entry tree. Thus word match equivalence is defined as in the Pre-wbe strategy, word matches are scored as in the Post-wbe strategy, and adjacent word scans are done as in the Pre-wbe strategy.

Each word match that extends beyond its kernel now has two scores, one up to the end of the kernel and a second representing the entire match. Individual word matches are ranked by their entire scores. The score of two consecutive word matches, however, must not be computed from their entire scores, because that would constitute scoring the region beyond the kernel twice. In order to allow the Control component to account for this overlap of adjacent word matches as it builds up multiword theories, the Matcher sends up both entire and overlap scores. Although performing two consecutive scans with this strategy requires more computation than previously, because paths in the entry tree are scored twice, word match reliability is substantially increased. Its efficiency and performance at the system level is better than either of the alternative strategies.

3. Use of Lexical Retrieval in HWIM

During the development of HWIM's Lexical Retrieval component, certain activities were codified that appeared especially useful within the system environment:

a) Global Scan

A sliding scan in both directions (left-to-right and right-to-left) done to establish a maximum score profile for shortfall scoring (see Sec. E, this volume), and in some cases to find seed words. If some region of the lattice is not suitably covered, subsequent scans are made in the vicinity of the uncovered region.

b) Left End Words Scan

A sequence of scans done at all possible left boundaries of the utterance looking for syntactically allowable initial words and then "near" the left end (as specified by Control) looking for possible seed words.

c) Word Anchored Scans

Scans done in the same direction as the word match with which they are to be phonologically compatible. (Sometimes Control has to make a very restricted scan, looking only for a word match going in the direction in which a subsequent extension is to be made.) The purpose of these scans is to make a phonologically consistent extension of the current theory in the given direction as specified by the word boundary rules.

d) End Scan

A scan used only for the Post-wbe strategy with Pre-wbe backup. Its purpose is to determine the best way to complete a utterance. This requires scoring all word boundary effects (as if doing an anchored scan from one specific word) and their associated ending penalties and then picking the best combination for use in scoring the complete utterance. (Ending penalties are determined by the APR component for each boundary in the segment lattice.)

4. Performance

We feel we have developed both an efficient and an effective method of lexical retrieval for use in general speech understanding systems. In order to demonstrate its stand-alone performance, it was tested on the same set of sentences that were used to evaluate the entire speech system. All entries in the expanded BIGDICT dictionary (1362) were possible candidates and only correct inflections were counted as correct. The performance is given for two different types of operation: a) sliding scans, and b) anchored scans.

a) Sliding Scan Performance

In this test, the Lexical Retrieval component scanned the segment lattice of each utterance for all dictionary words. A word match was counted as correct if it was a word in the utterance and was found in its proper place in the segment lattice. Although the number of words per sentence varied considerably, we made no attempt to compensate by requesting a proportional number of word matches. Instead, Lexical Retrieval was directed to return the 15 most probable word matches for each sentence. Figure 3 shows one performance measure: the average ratio of correct to incorrect words.

AVERAGE NUMBER OF WORDS PER SENTENCE	6.20
AVERAGE NUMBER OF DISTINCT WORD MATCHES . . .	12.30
AVERAGE NUMBER OF CORRECT WORDS PER SCAN . . .	2.17
AVERAGE NUMBER OF INCORRECT WORDS PER SCAN . .	10.13
AVERAGE RATIO OF CORRECT TO INCORRECT WORDS .	.2142

Fig. 3. Performance measurement.

Another performance measure derived from these same scans, the ability of Lexical Retrieval to distinguish between correct and incorrect words is presented in Figure 4. Here the rank of the highest scoring correct word is plotted as a histogram.

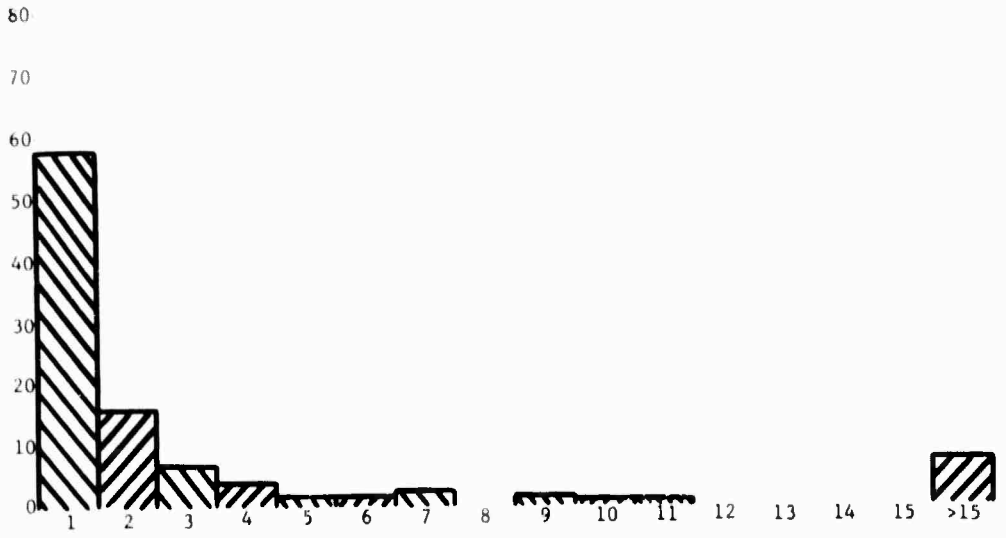


Figure 4.

b) Anchored Scan Performance

A more meaningful performance measure within the context of the whole system tested how well Lexical Retrieval performed when looking adjacent to the correct word in the correct place in the segment lattice. These scans were done as above with absolutely no selectional (i.e., syntactic class or word) constraints. We started at the beginning of each utterance and scanned for all words in the dictionary. Since only the top 15 positions were going to be used explicitly, scans were made with a short completion stack [Klovstad, 1976]. Occasionally, a second (selective) scan was necessary because the correct word would not be returned as one of the top scoring 15. Once the correct word was found, an anchored scan was made to its right. This procedure was continued until every word in the utterance had been found. In Figure 5, the rank (in the completion stack) of the correct word is plotted as a histogram. In this same figure, a cumulative distribution of these results is also presented. Almost all the "correct" words that were not among the top 15 were the result of noticeably bad segmentations in the lattice. For each anchored scan (321 were made in all), the score difference between the correct word and the highest scoring incorrect word was recorded. (N.B. This difference is positive only when the correct word appears in position 1.) The histogram of these score differences is shown in Figure 6.

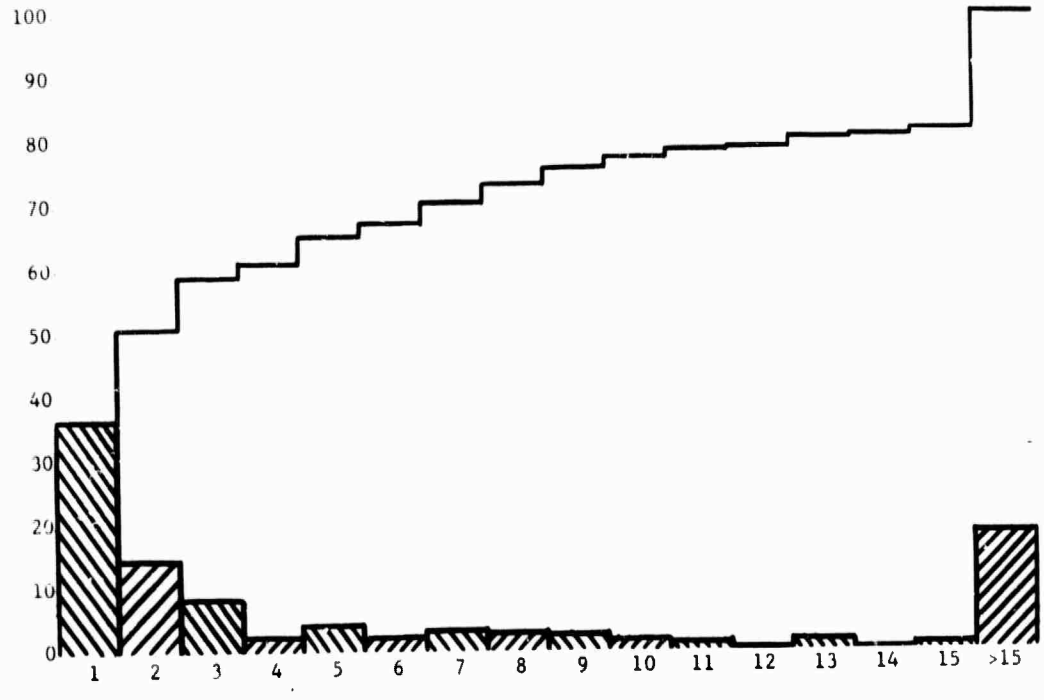


Figure 5.

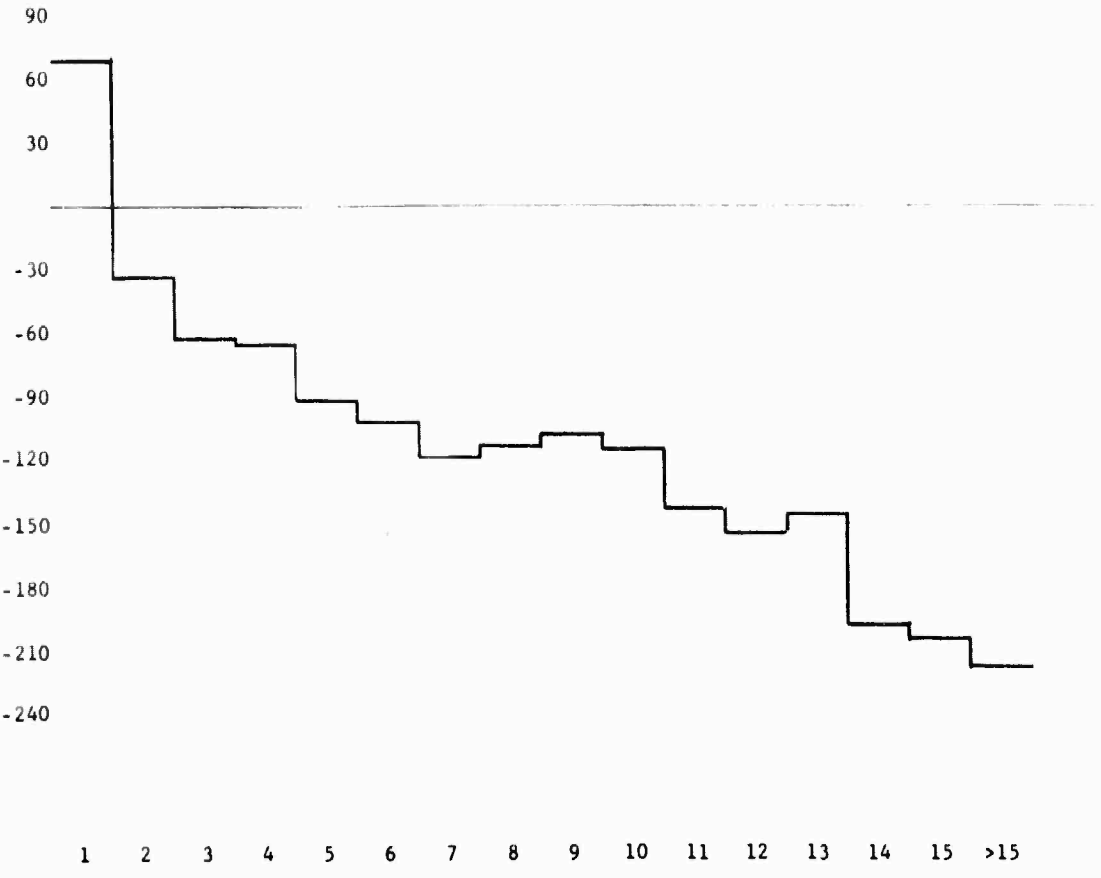


Figure 6.

E. CONTROL STRATEGY

1. First Attempts at A Control Strategy

In our original speech understanding system, SPEECHLIS, [Woods, 1974; Rovner et al., 1974], the control strategy was an implementation of the perceptual strategy described in Vol. I, Sec. B in its full generality. An initial scan of the segment lattice by the Lexical Retrieval component looking for robust cues was done to form initial one-word theories (stimulus driven hypothesization). Each such theory forced a call on a Semantic component to set semantic monitors, make proposals and detect events. The events and theories resulting from this semantic processing consisted of non-overlapping collections of semantically related words (generally separated by gaps). When the top-ranking theory became semantically "complete", at least with respect to the contents of the word lattice, a Syntactic component was called upon to evaluate it and propose words to fill the gaps.

The SPEECHLIS system contained separate queues of events, notices, and proposals, and was used experimentally to explore such issues as: when should proposals be done as opposed to doing another event, when should Syntax be called to evaluate a theory and make proposals, when should Semantics be called, and how dense should the initial word lattice be. This system was the culmination of our incremental simulation approach to speech understanding, and we learned a lot from it. As a result of our experience with this system, we evolved a more specialized framework for our second generation speech understanding system (HWIM), in which we embodied our best intuitions on these issues.

2. Island-driven Strategies

The type of strategy that appeared most effective as a result of our experience with SPEECHLIS is one that we have termed island driven. This is a strategy in which all theories consist of a contiguous sequence of words with no gaps between them (an island).

In the SPEECHLIS strategy, a theory generally contained several semantically-related islands separated by gaps. The reason for grouping

semantically-related islands into a single theory was the assumption that the semantic hypothesis that created such a theory could affect the syntactic processing of the individual islands. However, we failed to find useful ways to capitalize on this possibility, and in actuality, the syntactic processing of a given island was almost totally independent of other islands in the theory. (The only significant place where this was not true was when two islands grew close enough together that the gap between them could be filled by a single word. In this case, the Syntactic component took them both into account, making strong proposals in order to try to fill the gap.) On the other hand, the multiple island theories had an associated cost in processing due to the possibility of the same island occurring in several different theories (i.e., theories that differed in the words they hypothesized at other points of the utterance). Although care was taken that a given island need be syntactically evaluated only once, independent of how many theories it occurred in, a major portion of the Syntactic component's overhead in SPEECHLIS lay in attempting to keep track of islands that had previously been parsed and recognizing them when they appeared again in new theories. Thus, in handling multiple island theories, we were carrying a burden for which we were deriving at most a marginal theoretical advantage.

It should be noted here that a continuing goal in our search for effective speech understanding strategies has been to find ways for having what is found at one point in the utterance affect how we analyze other portions of the utterance. In most cases, syntactic information can only influence word hypotheses immediately adjacent to an island. The techniques explored in SPEECHLIS were an attempt to discover viable ways to get more global effects to lend support. However, while the semantic intersection technique does permit one to detect coincidences between widely separated, but semantically related, words, the way that this information was coupled into the Control and Syntactic components in SPEECHLIS did not provide sufficient support. A more effective way to tap this semantic intersection information might have been to use it in the priority scoring of theories, without explicitly forming combinations of separated islands. No further work was done in this direction, however.

3. Differences Between SPEECHLIS and HWIM Control Frameworks

In our second generation system, HWIM, various features of the general perceptual strategy described Vol. I, Sec. A were particularized, surrendering some flexibility for decreased overhead. On the other hand, we have implemented in HWIM a variety of flag-governed options, so that the effects of different control strategies could be explored. HWIM still represents only a sample point in an evolutionary process, and it still contains implementation overheads such as these flags which have been designed into it to promote flexibility for experimentation.

In both SPEECHLIS and HWIM, an initial scan of some portion of the utterance is used to form initial one-word seed events, but beyond this point, the two systems differ considerably. In HWIM, an event is processed by giving it to the Syntax component which combines syntactic, semantic, pragmatic, and prosodic information (see Vol. IV) to (a) determine whether the resulting theory could be part of a complete sentence; (b) produce proposals for adjacent words and/or categories; and (c) possibly adjust the score of the resulting theory to reflect syntactic, semantic, pragmatic, or prosodic information. Whereas in SPEECHLIS we had the option of doing several events before actively pursuing proposals, as well as that of monitoring all predictions while only actively proposing some of them, in HWIM we propose all predictions. Moreover, where SPEECHLIS gave Syntax the option of doing only a partial evaluation of a theory, in HWIM the Syntactic component does all its processing the first time it is called, making all possible compatible predictions at that time. This is because our experience with SPEECHLIS did not suggest any effective criteria for deciding when to call Syntax back for additional processing and predictions. Nor did it suggest a good criterion for calling for further unconstrained word scans in a portion of an utterance. As a result, monitors set without explicit proposals were only triggered by words from the initial scan or words independently proposed and serendipitously found.

Another major difference between the two systems is the short circuiting in HWIM of much of SPEECHLIS's monitor-propose-notice activity. Whereas SPEECHLIS permitted a number of events to be processed before any

of their proposals, in HWIM, proposals are done immediately as part of the processing of the event. Also, where SPEECHLIS used monitors to catch the words that came back from Lexical Retrieval and put them together with the theories that proposed them, in HWIM there is only one proposing theory for each batch of proposals. This means that appropriate notices can be constructed directly, eliminating the need for word lattice monitors (as well as one of the reasons for having a word lattice around!). This saves both the memory that had been required for storing these monitors and also considerable processing time.

Monitors remain in the HwIM system only for detecting island collisions, where two theories notice the same word from opposite directions. Whenever an event is created to add an additional word to a theory, an island table is consulted to see if the same word has been noticed from the other direction. If so, a collision event can be created that will combine the two theories and the noticed new word into a single new theory. Each entry in an island table is effectively a monitor that is watching for a theory to notice a particular word in a particular direction.

Other differences between HWIM and SPEECHLIS derive from (a) solutions to problems that had not yet been faced in SPEECHLIS; (b) the development of new knowledge sources (such as a synthesis-by-rule program and parametric matcher - Vol. II, Sec. C and D) and new versions of existing ones (such as a new parser and "pragmatic" grammar - Vol. IV, Sec. A and B); and (c) the systematization of a uniform scoring philosophy (Vol. I, Sec. B).

Under category (a) belongs the concept of rectification. In SPEECHLIS, we developed the concept of a fuzzy word match to accommodate finding several matches of the same word at approximately the same place, but with different end points and different scores. (This was due to several factors: different word boundary effects, branching in the segment lattice, and split and merge alignments in the Lexical Retrieval component (see Vol. III, Sec. D). However, we did not have a facility for checking the compatibility of word pairs from adjacent fuzzy word matches. Such a

facility would allow us to assign a "rectified score" to a theory which would take account of the amount by which the score of the best such pair falls below the sum of the scores on the best members of the two fuzzies. Thus, in SPEECHLIS, adjacent fuzzy word matches were given scores that might have been too lenient where adjacent word compatibility was concerned. In HWIM, techniques were developed to check this compatibility and assign scores accordingly.

4. Priority Scoring

The score assigned to a theory by the summation of lexical retrieval scores (essentially the log probability of its words being correct) we refer to as the quality score of the theory. We distinguish this from a possibly separate score called the priority score, which is used to rank events on the event queue to determine the order in which they are to be done. In early versions of HWIM, we used the quality score itself as the priority score. However, we have developed several algorithms with interesting theoretical properties using priority scores that are derived from but not identical with the quality score.

The first such score is a measure of the difference between the particular quality score for a theory and an upper bound on the possible quality score for any theory covering the same portion of the utterance. We call this difference the shortfall score, and it can be shown that using the shortfall score as a priority score under appropriate conditions guarantees finding the best scoring interpretation of the input utterance [Woods, 1976]. Using the quality score itself as a priority score does not guarantee this. Other priority scores are obtained by dividing either the quality score or the shortfall score by the time duration of the island to give quality density and shortfall density scoring, respectively. Since we have previously given a fairly complete derivation of the shortfall and shortfall density scoring strategies together with proofs of their completeness [Woods, 1976], we will instead present here a brief recapitulation of the different strategies.

5. Shortfall Scoring

As mentioned above, one of the kinds of priority scores that can be used for ordering the events on the event queue is a shortfall score. This score measures the amount by which the score of a theory falls below an upper bound on the possible score that could be achieved on the same region. When shortfall scoring is being used, a MAXSEG profile is constructed having the property that the score of a word match between boundaries i and j will be less than or equal to the area under the MAXSEG profile from i to j. (The latter is called the MAXSCORE for the region from i to j.) The shortfall score for a theory is then computed as the sum over all the word matches in the theory of the difference between the score of the word match and the MAXSCORE for the same region. This score is always negative, although only the magnitude is printed in the system traces (e.g. Vol. I, Appendix 2). The preferred theory is the one with the smallest absolute shortfall.

A MAXSEG profile is constructed incrementally as follows: Whenever the score of a word match exceeds the MAXSCORE for the region it covers, the excess score is distributed over the region to raise its MAXSCORE to equal the word match score. While the way in which this excess score is distributed does not affect the theoretical claims of the algorithm, it is desirable to do it in such a way as to minimize the amount by which the shortfall of other words that overlap the region is raised. Our current algorithm is to distribute the excess score over the segments covered by the word match that are not already bounded by the profile and to divide it proportional to the durations of the segments. Other distribution algorithms are possible, some of which have been tried. This one is better than some, but there are probably better strategies to be found. Keeping the MAXSEG profile as low as possible while still satisfying the upper bound condition is important, since excessively conservative upper bounds translate directly into an unnecessary increase in the breadth-first nature of the search, requiring more events to be processed before finding the chosen interpretation.

The theoretical characteristics of the shortfall scoring algorithm are such that if the words are returned by the Lexical Retrieval component in shortfall order and events are processed in order of increasing absolute shortfall (plus a few other assumptions, documented in [Woods, 1976]), then the first complete spanning interpretation found will be the best scoring interpretation that can be found by any strategy. We refer to this condition as "completeness". For speech understanding applications, completeness is a desirable property, but not necessarily essential if the cost of its attainment is too great. Shortfall scoring has the property of being complete without searching the entire space. Proof of its completeness depends only on the fact that when the first complete spanning theory is found, all other events on the queue will already have fallen below the ideal maximum score by a greater amount. Thus the result does not depend on the scores being likelihood ratios, nor does it make any assumption about the nature of the grammar (e.g., that it be a finite state Markov process), provided a parser exists that can make the necessary judgments. The completeness also does not depend on the order of scanning the utterance -- it is satisfied both for middle-out and for left-to-right strategies.

The actual implementation of the algorithm in the HWIM system does not satisfy the theoretical prerequisites exactly, since the words are not necessarily returned by the Lexical Retrieval component in shortfall order. However, a close approximation to the theoretical algorithm is achieved by retrieving words in relatively large batches and ordering them in shortfall order afterwards. An initial scan of the utterance is done to determine an initial estimate of the MAXSEG profile (as well as to find seed words), and this estimate is relatively stable thereafter. However, occasionally a word match is found in the course of an analysis that increases the MAXSEG profile in some region, and in that case all of the events whose scores could be affected are rescored and repositioned in the event queue. If a Lexical Retrieval component designed to return words in shortfall order were available, this would not be the case, and the initial upper bound would remain stable throughout the analysis (such a retrieval algorithm would have to determine a true upper bound in the course of its scan).

6. Density Scoring

Another type of priority scoring is density scoring. Here the score used to order the event queue is some basic score divided by the duration of the event. Conceptually, we can think of this priority scoring metric as predicting the potential score for the region not covered by a theory to be an extrapolation of the same score density already achieved. (In these terms, the shortfall strategy can be thought of as predicting that the upper bound will be achieved in the uncovered region.) Unlike the shortfall scores, density scores can get bad and then get better again as new words are added to a theory. Hence the density score is certainly not guaranteed to be an upper bound of the expected eventual score. However, it has another interesting property: in exactly those cases where it does not bound the eventual score, there is a potential seed word for the same ultimate theory somewhere else in the utterance that has a better score density and whose score density does bound the eventual score. This arises from the property of densities that the density of two regions combined will lie between the densities that they each have. It turns out that this alone is not sufficient to guarantee completeness for a density scoring strategy, since it is still possible for the density score starting from the best correct seed to fall below that of some other less-than-optimal spanning theory before it can be extended to a complete theory itself. However, with the addition of a facility for combining islands that start from separate seeds when they collide with each other, the density scoring strategy working middle-out from multiple seeds can be shown to be complete. Again, density scoring does not depend on any assumptions about the basic scores to which it is being applied other than that they be additive (and capable of division). Hence the density method can be applied to either the original quality score or to a shortfall score. The combination of the two methods in a shortfall density strategy seems to be more effective than shortfall or density scoring alone.

7. Ghosts

In the middle-out, island-driven strategies, each time an event is given to the Syntactic component for evaluation, proposals are returned and

word matches are found on both sides (except when the island is already at one end of the utterance). Although words are added to only one end of an island at a time, any theory that results from a given event will make proposals at the other end that are either the same as or a subset of the proposals that were made by its parent theory. Hence, it will eventually have to use a word at its other end that scores no better than the best word that was found there at the time the event was noticed.

The HWIM system, under the control of a flag, can keep track of the words that were found at the other end of an island when an event is created. We refer to these words as the ghosts for the event. By computing the score of an event using the best of the ghosts as well as the words of the event itself, we can gain an effective lookahead of one word in the opposite direction. This promotes the acceleration of good events up the queue and bad events downwards. We also remember the proposals used to find the ghosts and if the extension of the event does not in fact result in tightening those proposals, then the ghost words can be used again for the new theory without having to re-call the Lexical Retrieval component.

8. Choosing Direction

As mentioned above, whenever an event is processed, new words are generally found at both ends of the new island. Any spanning theory that can eventually be derived from such an island can be reached either by adding the left word first and continuing to process the resulting theory, or by adding the right word first and proceeding from there. This results in a potential combinatoric explosion, since in general a theory of k words can be derived in 2^{k-1} different ways. This explosion can be limited, however, by testing every event that is noticed to see if the theory it produces has been found before. The problem could be eliminated completely by choosing arbitrarily to use only the left notices or the right notices (the choice can be made to depend on aspects of the particular situation such as the direction having the best scoring new word or the shortest distance to the end of the sentence). While this option is available in the system under control of a flag, there are advantages to pursuing both directions when the Lexical Retrieval component is given a limit on the

number of words that it may retrieve. In this situation, it is possible that extending an island in a given direction can result in tighter predictions at the other end and may make it possible to find a word there that was buried in the noise before. The CHOOSEDIR option gains this benefit while effectively controlling the combinatorics.

Under the control of a flag called CHOOSEDIR, HWIM will check each time an event is processed to see if it is the first successful event resulting from its parent. If so, HWIM marks that direction as the chosen one for all events stemming from that parent, and rescores all related events going in the opposite direction using the worst of the ghost words instead of the best. The effect of this rescoring is to demote all of the events going opposite to the chosen direction by a score equivalent to the difference between the scores of the best and worst ghosts. If the proposals used in retrieving those ghosts are relatively lax, or the score of the worst ghost is sufficiently high that calling the Retrieval component back with tighter predictions might result in new words with reasonable scores, then this demotion is not severe. Otherwise it effectively rules these events out of consideration. The effect is that "wrong way" events are considered only if they have a chance of producing new words with a reasonable score. Moreover, they are ranked in the event queue at the point where they will be processed only when a word with such a score would be the best thing to do. All of these wrong way events are blocked from noticing any of the words in their ghost list, since all those words have already been pursued in the chosen direction. This feature gains almost all of the combinatoric control that choosing only one direction would gain, while giving the potential for using tighter predictions to find new words with exactly the right priority ranking in the event queue. The use of ghosts and the chosen direction heuristics do not affect the completeness of strategies that would otherwise be complete.

9. Other Components of Score

In addition to the basic lexical score described above and the various priority scores that can be derived from it, there are three other components of an event score. These are an ending score assigned by the

APR component to boundaries where an utterance could begin or end, a verification score, and a syntactic score. Ending scores are zero except when a theory hypothesizes an end at a boundary that the APR has specified as possible but unlikely. Verification scores are assigned by the Verification component based on a parametric matching between hypothesized words and the input waveform. They are calibrated statistically as described in Vol. I., Sec. B and Vol. II, Sec. D to produce log likelihood ratios that are compatible with the Lexical Retrieval scores. Syntactic scores reflect penalties for linguistic inconsistencies that decrease the likelihood of an interpretation but do not rule it out altogether; they are also in the form of log likelihood ratios. In all of the experiments that we have so far run, the syntactic scores have always been zero. Mechanisms have been set up to compare prosodic cues in the signal against prosodic expectations in the grammar and return the result as a component of the syntactic score (Vol. IV, Sec. C), but we have not been able to run experiments to test this facility. All three of these "non-lexical" scores are converted to score densities when one of the density scoring strategies is being used. The syntactic and verification scores (like the lexical match score) are divided by the duration of the island; the ending score is divided by the duration of the utterance. The use of verification scores and syntactic scores can affect the theoretical completeness of a strategy that is otherwise complete.

10. Control Strategy Options

The selection of a particular strategy in HWIM is determined by the settings of global flags that govern various strategy options. In the final version of the system there are 25 such flags, with a total number of permissible combinations exceeding 5000. In this section we will describe the major such flags. Other flags and other control variables govern such things as a limit on the number of events to be processed before giving up, stack lengths for various requests to Lexical Retrieval, and a variety of experimental control strategies.

Major Control Flags:

a) PRONLIKEFLAG

The phonlogical rule component includes a facility for taking pronunciation likelihood estimates from the base-form pronunciations of words and combining them with likelihood multipliers indicated in the optional phonological rules [Woods et al., 1975] to produce likelihood estimates for each pronunciation. The Lexical Retrieval component returns these likelihoods, expressed as log likelihood ratios, with each word match that it finds, but keeps this number separate from the lexical score so that the Control component can decide whether to use it or not. When PRONLIKEFLAG is T, Control uses the lexical score plus the pronunciation likelihood as the quality score for the word. When PRONLIKEFLAG is NIL, only the lexical score is used. In all of the experiments described in this report (Sec. F; Vol. I, Sec. D), PRONLIKEFLAG is T.

b) SHORTFALLFLAG

As discussed above, one of the strategies involves using a shortfall score instead of the straight quality score for ordering the queue of events to be done. When SHORTFALLFLAG is T, a MAXSEG profile is constructed and the priority score of an event is computed using the sum of the differences between the the actual score and the MAXSCORE of each word match involved.

c) DENSITYFLAG

When the DENSITYFLAG is set, durations are carried with every event, and event scores are computed using either the quality or shortfall scores (depending on SHORTFALLFLAG) divided by this duration. Individual word matches in fuzzy word matches are also sorted by density order in this case, and the non-lexical components of the score (end penalties, verification scores, and syntactic scores) are divided by durations (as described above) to make them comparable.

d) VERIFYFLAG

VERIFYFLAG has four possible values: NIL, VERIFY@NOTICE, VERIFY@DO, and VERIFY@PICK. The first calls for no verification at all. The second indicates that the Verifier should be called when new words are noticed, so that the score assigned to the event includes the verification score of the new word. VERIFY@DO, on the other hand, indicates that verification should wait until the time that the event is processed. Thus the score of an event in the event queue does not include the verification score of the new word, but does so for all of the words in the parent theory. VERIFY@DO results in fewer calls to the Verification component, but does not use the verification information as early as possible.

VERIFY@PICK provides the best features of the other two options. It gives each event a default verification credit at the time it is noticed, which is comparable to the best possible verification score (the amount of score to default is specified by a control variable). Then, at the time when the event reaches the top of the event queue and is about to be processed, Verification is called for the new word to replace the default score by a real one. If the rescored event is still the best scoring event, then it is processed. Otherwise it is re-inserted at the appropriate point in the queue, and the process is repeated for the new top item in the queue. The system can tell the difference between an event that includes a default verification score and one that has been verified, so that when an already verified event reaches the top of the queue it will not be verified again. VERIFY@PICK thus effectively uses the verification scores of all of the words in an event in deciding which event is to be processed (gaining the advantage of the VERIFY@NOTICE strategy), while actually calling the Verification component for only those events that make it to the top of the queue (the advantage of the VERIFY@DO option).

e) HYBRIDFLAG

HYBRIDFLAG has three possible values: NIL, 0, and a non-zero integer. When HYBRIDFLAG is zero, the initial scan of the utterance for seed words consists only of words beginning at boundaries that the APR component has labeled as possible left ends. When it is an integer greater than zero, the initial scan is permitted to go beyond this point by a number of frames (10 millisecond intervals) equal to the value of HYBRIDFLAG. In either case, when HYBRIDFLAG is non-NIL, theories are not permitted to notice words on the right until they have been extended to the left end of the utterance and the left end event has been done. Hybrid strategies permit skipping the first word or two of the utterance in order to find a robust anchor word, but do so for at most a bounded distance into the utterance. They shift to left-to-right processing as soon as possible and appear to have a good combination of the advantages of both left-to-right and middle-out strategies.

f) GHOSTFLAG

GHOSTFLAG causes the control component to include with each event a record of the words that were noticed at the other end of the theory when the event was created. The best such ghost word is then used in addition to the words of the theory and the new word to determine the score of the event.

g) CHOOSEDIRFLAG

When CHOOSEDIRFLAG is T, the direction choosing strategy [Sec. E.8] is enabled. That is, when the first successful event for a given parent theory is found, all events spawned by that parent going in the opposite direction are rescored using their worst ghost word and are blocked from noticing any of their ghost words. Their only function is to try to find additional words at the chosen end by making more specific proposals resulting from larger context. CHOOSEDIRFLAG can only be enabled when GHOSTFLAG is set.

h) COLLISIONFLAG

COLLISIONFLAG enables the island collision mechanism. When it is on, each event is entered into one of two island tables depending on the direction of the event. The other island table is then consulted to determine if the same word at approximately the same place has been noticed in the other direction. If so, a collision event is created, which when processed will combine the two noticing theories and the new word between them into a single theory. This event is scored using all of the words of the two theories plus the new word (plus the ghosts of both colliding events if GHOSTFLAG is on).

11. The 12 October Strategies

As mentioned earlier, all of HWIM's control strategy options fall within a general island-driven framework. They each perform an initial scan of some region of the utterance, creating initial one-word seed events. In general "middle-out" strategies, the initial scan is done over the entire utterance. In "left-to-right"" strategies, the initial scan only considers words that could begin the utterance (i.e., whose left boundary falls at one of several possible points that have been labeled by the APR component as possible starting points of the utterance). In "hybrid" strategies, the initial scan is done on a fixed initial portion of

the utterance (from its beginning to some point a specified number of milliseconds later). Then a middle-out analysis is done on this region with the remainder necessarily being analyzed left-to-right. In all of these strategy options, events are ordered on the queue by their priority scores. The choice of a particular combination of strategy options is determined by the settings of the control flags described above. In this section, we will describe some particular combinations that we have used for testing the final speech understanding system as of October 1976.

The preferred algorithm, which we chose on the basis of preliminary experiments to use for our final test run, is the shortfall density strategy running in hybrid mode with an initial hybrid search displacement of 0. (We refer to such a hybrid strategy as "left hybrid".) The strategy uses the Verification component with the VERIFY@PICK option, and includes the use of ghosts although they don't contribute anything except for a few seeds not immediately at the left end. As a convenient mnemonic for naming this and other strategies, we use the abbreviations LH for left hybrid, SD for shortfall density, and VP for verify at pick. The strategy just described is thus refered to a LHSDVP. The corresponding middle-out strategy, not using the hybrid option would be simply SDVP. Alternatives to SD are S for shortfall alone, Q for quality alone, and QD for quality density. Alternatives for VP are VD for verify at do, VN for verify at notice, and NV for no verify.

The strategies compared in experiments with the final system include:

a) LHxNV for x = S, SD, Q, QD
i.e., Left hybrid strategy without Verification, using the different priority scoring metrics--S, SD, Q and QD.

b) xNV for x = S, SD, Q, QD
i.e., Middle out strategy without Verification using different priority scoring metrics. Ghosts, island collisions, and direction choosing heuristics are all used in these experiments.

c) LHSDx for x = VP, VN, VD, NV
i.e., Left hybrid shortfall density using various Verification options.

d) SD+x for x = 0, C, G, GD, GDC
i.e., shortfall density without Verification using different combinations of the ghosts, island collision, and direction choosing heuristics. SD+0 uses none of the three options, C stands for island collisions, G for ghosts, and D for direction choosing (CHOOSEDIR).

F. SYSTEM PERFORMANCE

The performance of the latest (12 October) version of HWIM was demonstrated by running it with vocabularies of two sizes on a test set of 124 sentence-length utterances, as described in Vol. I, Sec. D. To recapitulate, these results are shown in Table 1 below.

	BIGDICT (1097 words)	MIDDICT (409 words)
Correctly understood	54 = 44%	65 = 52%
All words correct	51 = 41%	59 = 48%
Semantically correct	3 = 2%	6 = 4%
Incorrect	45 = 36%	40 = 32%
Close to correct	29 = 23%	25 = 20%
Less correct	13 = 10%	10 = 8%
Very wrong	3 = 2%	5 = 4%
No response	25 = 20%	19 = 15%
Gave up (150 theories)	24 = 19%	18 = 14%
System broke	1 = 1%	1 = 1%
Estimated average branching (words)	196	67
Speed (times real time)	1350	1050

Table 1. Summary of final performance results, 124 utterances by three speakers.

- - - - - - - - - - - - - - -

The test sentences are listed in Vol. 1, Appendix 1, and the performance results for each test utterance are listed in Appendix 3, this volume. This section discusses additional aspects of system performance - speed, speaker variation, and the results of some experiments in strategy variation performed with a small subset of the 124 utterances.

1. Real Time Issues

One of the objectives of the ARPA Speech Understanding research program was to discover and develop techniques that would permit the understanding of large-vocabulary, continuous speech in a few times real time on a 100 mips (millions of instructions per second) machine. This target was established on the grounds that the machines on which speech understanding systems would eventually run would be much faster than the machines available at the initiation of the project in 1971. By way of

comparison, the DEC PDP-10 KA processor on which the BBN speech understanding work has been done is approximately a .34 mips machine [Fuller, 1976]. This is a relatively slow machine by today's standards, but one on which a great deal of sophisticated software has been developed.

For a research project such as this one, the prior existence of the software necessary for experimental system development is far more important than basic machine speed, although other factors being equal, a faster machine is clearly preferable. In our case, the existence of an interactive INTERLISP system and the flexibility of the TENEX file and process structure -- especially the multiple process capabilities -- have contributed significantly to the project. The development of such facilities to take maximum advantage of a new machine may require several years. Since the five year time period set out for the speech understanding program was considered barely enough time to do the necessary experimental development of the speech systems themselves, it was not feasible to work in the context of untested machines and operating systems. This decision meant that the experimental systems that we developed would be considerably slower than would be possible by running on appropriate hardware configurations.

In addition to this basic factor of machine speed, we elected to accept an additional inflation in the running time of our own system in order to gain the flexibility for experimentation and system reconfigurability that comes from using high-level programming languages such as INTERLISP and from using styles of programming that promote changeability at the cost of run-time efficiency. For example, INTERLISP spends extra instructions checking data types and array bounds, a procedure that greatly reduces the debugging cost. In addition, extra instructions are spent 1) in keeping records of the current state of a running program so that they will be available for inspection by a programmer at any point in the process; and 2) in testing flags to determine which options to use at various points in the programs, so that experimental comparisons of different strategies can be easily carried out. Trace instructions were also embedded at significant points in the code so that comprehensive records of the intermediate stages of the understanding process could be

kept and studied (an example of such a trace is shown in Vol. I, Appendix 2).

Given that today's experimental speech understanding systems run significantly slower than ones that would be constructed for applications, a question of interest is what speeds would be possible if the techniques used in the BBN speech understanding system were appropriately implemented on suitable hardware?

We will assume that the signal processing, which requires approximately 55 times real time on our current facility, can be done in real time either with analog circuitry, or with fast digital processors. Similar kinds of processing are already being done in real time on such processors as the SPS-41 and the Lincoln Laboratory DVT [Hofstetter et al., 1975]. The remainder of the system runs in approximately 1300 times real time, that is, the number of seconds of computer time required to understand an utterance is approximately 1300 times the duration of the utterance in seconds. Some utterances can be understood in less time, and some require longer, but this is an approximate average.

Of the total time required by the non-signal processing parts of the system, the Acoustic-Phonetic Recognition (APR) component takes a negligible amount of time, and the remainder is broken down roughly as shown below. (Here "ss" stands for "second of speech.")

Component	Percentage	CPU sec./ss
Lexical retrieval	40%	520
Verification	15%	195
Control and Syntax	40%	520
Tracing	5%	65

(The figure given for tracing time is that of a single function PRINTEVENTQ, which prints the event queue, the most time-consuming single tracing function. There are many other tracing functions embedded throughout the Control and Syntax components, so the actual trace time is somewhat higher than indicated and the Control and Syntax time is therefore somewhat less than indicated. Of course, in a non-experimental system, this tracing would not be done.)

One of the overheads of jobs running on TENEX is due to the demand paging access to a virtual memory that TENEX provides. Each running process (called a "fork") has a virtual memory of 262,144 words, which is mapped into a working set of in-core pages, each consisting of 512 words. Whenever a running process accesses a memory location that is not in core, the process waits while the operating system fetches the necessary page into core memory, possibly removing another page from the working set to make room. This procedure is called a "page fault" and the operating system charges the process for running the "pager trap code" that determines how to swap the required page into core. This time is included in all of the system's clock times and is impossible to separate out from the time required for actual instructions in the running process. When a system is lightly loaded and an individual process is permitted to accumulate a large working set in core, the number of page faults can be few, but when the load is heavy and the process is swapped out frequently, a process can spend significant amounts of page fault time retrieving the pages it has lost. To make matters worse, when one fork calls another, the pages of the idle calling fork are made available for reuse at the same time that the called fork is demanding pages, so that there is likely to be significant page contention between two "cooperating" forks. Since the speech understanding system contains a number of forks that call each other, this effect is significant.

The upshot of the above discussion is that there is a largely uncontrollable, variable, and not adequately measurable component due to page faulting in all of the times that we have measured for our system. If the system were running in an environment where the necessary number of pages of real memory were available to the processor at all times, all of this time would go away. Average TENEX system performance shows that about 12% of the "sold" time (time charged to user processes) is spent handling pager traps. The HWIM system, with its multiple INTERLISP forks and large data structures, can be expected to incur page faults more often than the average; it seems safe to estimate that something like 15% of the CPU time measured by the system is page faulting time (and the actual amount may be significantly more).

Major improvements in time requirements can be gained by reprogramming several of the components for speed. This has been done to some extent already with the Lexical Retrieval component, which has been written in BCPL, a language that compiles into fairly efficient machine code. However, the Lexical Retrieval component currently includes overhead for a variety of options we are not now using, and writing in BCPL does not give one as efficient access to the accumulators of the machine as would machine language. We estimate a factor of two could be gained from eliminating unused parts of the code and by recoding from BCPL into machine language.

The Control and Syntax components of the system are both implemented in INTERLISP, a language that gives great flexibility in experimental program development, while charging a fairly high penalty in overhead. For example, in the timing breakdown printed on the traces, a very low level function, SHORTORDERP, was found to take about 4 milliseconds per call. All this function does is test a global flag to determine which of two comparisons to make and then compare two floating point numbers accessed from an array to determine which is greater. It is almost an understatement to say that the effect of this function can easily be done in less than 100 microseconds. (The rest of the time is spent in creating INTERLISP stack entries, adding the names of the variables so that they will be available for subsequent stack scans, checking that the subfunctions called are indeed defined functions and that the arrays accessed are defined arrays and the indices are within bounds, etc.) This would indicate that a factor of at least 40 in speed could be gained in reprogramming the INTERLISP portions of the system into machine language. Of course, this is just one function, but it is executed in one of the innermost loops of the system. Previous experience with translating INTERLISP code into machine language indicates that speedups of this magnitude are indeed achievable. Speedups of a factor of 10 have been achieved even within INTERLISP by block compiling groups of functions together, and this has not yet been done to the code in question.

The CPU times measured for the Control and Syntax components also include the setup times for the Syntax and TRIP forks. These setup times should not have been included in the run time measurements, but were done

so automatically, given the way we had implemented the functions. Hence, the measured times for the Control component are high by this amount (approximately 90 seconds of CPU time per utterance, regardless of duration). Taking everything into account, we feel safe in projecting a factor of 40 speedup in the Control and Syntax components by reprogramming to machine language.

The Verification component is already implemented in a combination of Fortran, BCPL, and machine language. Further gains in its efficiency are expected to be small.

Summarizing the above discussion, we would project the following times to be realizable on our current computer configuration by reprogramming individual components for speed:

Component	Current CPU/ss	Projected CPU/ss
Lexical retrieval	520	260
Verification	195	195
Control & syntax	520	13
Tracing	65	0
	1300	468

If this were implemented on a dedicated PDP-10 with sufficient core memory to eliminate page faulting (or make it insignificant), an additional improvement of at least 15% would result, bringing the total time requirements to 400 CPU seconds per second of speech (on a .34 mips machine).

On a machine with the capability of 100 mips (an instruction time of 10 nanoseconds), this would require only 1.36 CPU seconds per second of speech (or 1.36 times real time), thus exceeding the projected goals of "a few times real time".

All of the above discussion assumes a single serial processor with the speeds indicated. However, the speech understanding algorithms developed appear to lend themselves well to parallel implementation. At the control level, the task of understanding a sentence consists of a search in a space whose depth is the number of words in the utterance, but whose branching is a function of the competition from other partial theories that are acoustically similar. We typically process on the order of 100 events to

determine the chosen interpretation, but only on the order of 10 of them (i.e., the number of words in the utterance plus two end events) are actual steps in the construction of the chosen theory. The others are competitors that were evaluated to determine which theory was best. Thus it looks like almost a factor of 10 in parallelism might be achieved for our current system, by using a configuration of 10 processors sharing a common memory. CMU has been exploring such parallel-processor implementations of speech understanding systems. With such configurations, the necessary 136 mips for real time operation could be achieved using processors with 75 nanosecond instruction times.

All of the above discussion assumes no basic change in the algorithms or performance of the current system. Undoubtedly there are improvements that remain to be discovered in the algorithms that would yield further speed improvement. Moreover, a major factor not considered in the above analysis is the effect of segmentation errors and gross mislabelings by the APR component on the overall number of events that must be processed to determine the chosen interpretation. Current experience indicates that, due to a cascade effect through the various processing levels, an error in segmenting or labeling at the APR level can give rise to a number of extra events being processed at the control level before the correct interpretation emerges (if it does so before an incorrect interpretation is discovered). The number of such events depends on the severity of the original error in a non-linear way so that doubling the severity of an error more than doubles the number of extra events to be processed. Consequently, a small decrease in the number and severity of APR errors can result in a significant decrease in the number of events processed. Thus, an improvement in APR performance statistics alone can significantly improve the overall speed of the system independent of any implementation efficiency improvements. We can therefore expect that continued extension of the range of acoustic-phonetic events that the APR component recognizes and handles correctly would result in an additional speedup of the system. A factor of two here would not be unreasonable to expect.

Cost Issues

Another relevant question, especially given the previous discussion of machine speeds and multiprocessor configurations, is the expected cost of such systems. One might expect that such a machine would be a giant configuration costing millions of dollars. However, the costs of computer processors and memory are currently coming down dramatically in conjunction with speed improvements. In a study done at the Harvard Business School, John Doerr [1976] estimates that by 1980 a 10 MIPS system suitable for a speech understanding system could be built for about $25,000. His system consists of a $500 CPU, $3,600 fast primary memory, and $22,500 secondary memory. At this rate, a 10 processor configuration sharing the secondary memory would cost only $63,500 and would have an effective capacity of 100 mips, for the BBN speech understanding task.

2. Performance vs. Speaker

The system performance for each speaker is shown in Table 2 below.

	BIGDICT			TRAVELDICT		
	WAW	JJW	RMS	WAW	JJW	RMS
Correct	30	13	11	36	16	13
Incorrect	20	9	16	20	7	13
No response	12	9	4	6	8	5

Table 2. Number of sentences in 3 response categories, broken down by speaker.

Although there are differences across speakers, a chi-square test shows that the differences in each case are significant only at a level of about p=0.25. That is, there is no strong evidence that the observed differences across speakers did not occur by chance.

3. Accuracy vs. Sentence Length

The "correctly understood" performance (of the BIGDICT system) vs. utterance length in words and duration in seconds is shown in Tables 3 and 4 below:

Words	Accuracy
3	1/8 = .12
4	6/14 = .43
5	18/30 = .60
6	13/24 = .54
7	8/20 = .40
8	6/18 = .33
9	0/2 = 0.
10	2/6 = .33
13	0/2 = 0.

Table 3. Accuracy vs. utterance length.

Duration (sec)	Accuracy
0.7 - 1.0	1/6 = .17
1.0 - 1.2	7/10 = .70
1.4	6/14 = .43
1.6	6/15 = .40
1.8	12/22 = .55
2.0	8/17 = .47
2.2	5/15 = .33
2.4	4/7 = .57
2.6	3/7 = .42
2.8	2/6 = .33
2.8 - 4.0	0/5 = 0.

Table 4. Accuracy vs. utterance duration.

(In Table 3, "words" are counted as lexical entries as shown in Appendix 5. Besides standard English words, there are a few compound adjectives (ONE-WAY, ROUND-TRIP) and place names (EL@PASO), and the plural morpheme is a separate word (Bill's = BILL -S).)

Table 3 shows that the test set contained relatively few utterances shorter than 4 or longer than 8 words long. The other data points are not sufficient to allow one to make any quantitative conclusions about the nature of the decline in performance as the length of the sentence grows. The 3- and 4-word data points are of some surprise, since one might expect there to be fewer chances for the system to go wrong with such short sentences.

It would seem that the 3-word sentences, and possibly the 4-word ones as well, may be spoken differently than longer ones. Sentences like "Print her fare" and "Figure his expenses" seem to lack the flow of the longer ones like "Schedule a trip by train to New York." They sound more like a series of isolated words, and this may affect their acoustic-phonetic characteristics.

Furthermore, there are "ending effects" that should be relatively more important for a short sentence than for a long one. For example, the ends of the utterance, where the speech begins with maximum subglottal pressure, and also where the speech ends, are regions of rather different acoustic-phonetic characteristics. Also, in a short sentence, there is less syntactic context and hence less syntactic constraint at each choice point. Such effects may swamp the high expectations of a monotonic performance vs. length model.

Much the same can be said about the performance vs. duration data. A rather larger set of test utterances than we had the resources to process would have been necessary to explore these questions more fully.

4. Experiments in Strategy Variations

The "LHSDVP" control strategy used in the final (12 October) version of HWIM for the performance tests described above represents only one of a large number of strategy variations that are controlled by the values of about 25 flag variables in the Control component. The selection of this particular strategy was based on experimentation on the effects of the various heuristics on system performance during the development of the Control component. At the end of the project, it would have been instructive to have tested many strategy variations on a large test set of utterances in order to evaluate their effects quantitatively, as has been done by Paxton [1976]. However, time limitations prevented us from doing that with a test set as large as 124 utterances.

As a compromise, a set of 10 utterances was selected randomly from the set of 124. Four strategy-variation experiments were performed by running that smaller set of utterances on each of 14 versions of HWIM. Such a small test set cannot give significant quantitative results, but it is hoped that it will give the "flavor" of the heuristics involved. The per-utterance results from these four experiments are presented in Appendix 4 of this volume. The remainder of this section briefly discusses those results.

Experiment 1: LHxNV

This experiment used the same left-hybrid strategy as the 12 October final version of HWIM, but without Verification; the experimental variable, x, was the method of scoring hypotheses - either quality (Q i.e., LHQNV), quality density (QD - LHQDNV), shortfall (S - LHSNV), or shortfall density (SD - LHSDNV). In all four of these strategy variation experiments, the "give up" threshold was set to 100 theories rather than 150.

As can be seen in Appendix 4, two utterances yield no useful comparisons, because for all four methods, processing was terminated before a spanning theory was formed. We can summarize the results as shown below. (The row labeled NTH5 gives the total number of theories required for the five utterances understood correctly by methods Q, QD, and SD.)

	LHQNV	LHQDNV	LHSNV	LHSDNV
Correct	6	5	0	6
Incorrect	1	1	0	2
No response	3	4	10	2
NTH5	145	268	-	183

Shortfall scoring (S) is so breadth-first that no sentence was understood in as few as 100 theories. Quality density (QD) is also significantly more breadth-first than the remaining two, as shown by its value of NTH5 and the larger number of utterances it failed to finisn. Quality and shortfall density get the same number correct, with quality requiring fewer theories. We should expect quality scoring strategies to reach completion faster, but we also expect it to be led astray more easily. The sampling effect of the small number of utterances in this experiment does not let us judge this.

Experiment 2: xNV

This experiment used the more general middle-out strategy (including the collision, ghost, and [illegible]OSEDIR heuristics) described in Sec. E; again, the experimental variable was the scoring method.

In this case, three utterances were not completed and therefore provided no useful comparisons. The results are summarized below. (The row labeled NTH3 gives the sum of the number of theories understood correctly by methods Q, QD, and SD.)

	QNV	QDNV	SNV	SDNV
Correct	4	3	0	5
Incorrect	2	0	0	0
No response	4	7	10	5
NTH3	23	142	-	96

Again, shortfall scoring produced no completions, with quality density significantly more breadth-first than quality and shortfall density. In this case, where we can compare quality and shortfall density directly, quality was much faster, but it was not as accurate.

These two experiments allow us to compare left-hybrid and middle-out strategies. The results for the more successful scoring methods, quality and shortfall density, are shown below. (As before, NTH4 is the summed theory count for the four utterances correctly understood by both.)

	LHQNV	QNV		LHSDNV	SDNV
Correct	6	4		6	5
Incorrect	1	2		2	0
No response	3	4		2	5
NTH4	67	34	NTH5	154	255

In both cases, the left-hybrid strategy appears to get more utterances correct, but in terms of number of theories required, the more general middle-out strategy appears to be more efficient in the case of quality scoring. If that effect is real, and not an artifact of our small sample, then we can offer no rationalization for it.

Experiment 3: SD+x

This experiment also used the middle-out strategy with shortfall density scoring and no verification; the variable was the inclusion of the three heuristics called collision events (C), ghosts (G), and CHOOSEDIR (D), which are described in Sec. E above. There are five, rather than eight, combinations, since CHOOSEDIR requires the ghosts heuristic to be operating.

The results are summarized below. (NTH3 denotes the summed theory count for the three utterances correctly understood by all methods.)

	SD+0	SD+C	SD+G	SD+GD	SD+GDC
Correct	3	3	3	4	5
Incorrect	0	0	0	0	0
No response	7	7	7	6	5
NTH3	162	121	145	130	96

Examination of the individual utterance results in Appendix 4 shows that the inclusion of a heuristic does not always guarantee that the system will understand the utterance in fewer theories, but the pooled results above (note especially the series SD+0, SD+G, SD+GD, SD+GDC) suggest that the successively added heuristics produce improvements in both accuracy and number of theories required.

Experiment 4: LHSDxV

This experiment used the left-hybrid strategy with shortfall density scoring; the variable was the strategy for using the Verification component. The strategies, which are described in Sec. E, are LHSDNV (No Verify), LHSDVD (Verify-at-Do), and LHSDVP (Verify-at-Pick). (One strategy, Verify-at-Notice, was not run because of lack of time; its performance should have been identical to that of Verify-at-Pick, with higher run times due to many additional calls on the Verifier.)

Each method resulted in six utterances correctly understood, two incorrect, and two no response. The interesting results are the number of theories and the CPU times required for the six correctly understood utterances.

	LHSDNV	LHSDVD	LHSDVP
Theories	183	141	100
Run time (sec.)	9443	9156	8276

Even though this small test set shows no differences in accuracy, it does show that Verification improves HWIM's performance. The CPU time used by Verification is considerable (about 5 sec. per word verified), but the decrease in the number of theories more than compensates for the extra processing cost. The comparison of LHSDVD and LHSDVP shows the advantage of applying Verification as early in the process as possible. The extra calls to Verification in LHSDVP result in a better ordering of the event queue, resulting in fewer incorrect events being processed and a net saving of processing time.

References

[1] Bobrow, D. and B. Fraser (1968)
"A Phonological Rule Tester", CACM, Vol. 11, No. 11, November 1968.

[2] Chomsky, N., and M. Halle (1968)
The Sound Pattern of English, Harper and Row, New York NY.

[3] Doerr, J. (1976)
"The Commercial Feasibility of Speech Understanding." Unpublished research report, Harvard Business School, Harvard University.

[4] Fuller, S.H. (1976)
"Price/Performance Comparison of C.mmp and the PDP-10," Third Annual Symposium on Computer Architecture, Conference Proceedings. Vol. 4, No. 4, January 1976 Computer Architecture News (ACM-SIGARCH), pp. 195-202.

[5] Hofstetter, E.M. et al. (1975)
"Vocoder Implementations on the Lincoln Digital Voice Terminal", EASCON '75, Washington D.C.

[6] Kenyon, J.S. and T.A. Knotts (1944)
A Pronouncing Dictionary of American English, G.C. Merriam Co., Publisher, Springfield MA.

[7] Klatt, D.H. and K.N. Stevens (1973)
"On the Automatic Recognition of Continuous Speech: Implications from a Spectrogram-Reading Experiment," IEEE Trans. on Audio and Electroacoustics, Vol AU-21, No. 3, June, pp. 210-217.

[8] Klovstad, J.W. (1976)
"Probabilistic Lexical Retrieval with Embedded Phonological Word Boundary Rules," in Woods et al., (1976b).

[9] Klovstad, J.W. and L. Mondschein (1975)
"The CASPERS Linguistic Analysis System," IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-23, No. 1.

[10] Nash-Webber, B.L. (1975)
"The Role of Semantics in Automatic Speech Understanding," in Representation and Understanding: Studies in Cognitive Science, D. Bobrow and A. Collins (eds.), Academic Press.

[11] Newell, A. et al. (1973)
Speech Understanding Systems: Final Report of a Study Group, North-Holland/American Elsevier.

[12] Oshika, B.T., V.W. Zue, R.V. Weeks, H. Neu and J. Aurbach (1975)
"The Role of Phonological Rules in Speech Understanding Research," IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-23, No.1.

[13] Paxton, W.H. (1976)
"Experiments in Speech Understanding System Control," Artificial Intelligence Center Technical Note 134, Stanford Research Institute, Menlo Park, Ca.

[14] Rovner P., B.L. Nash-Webber and W.A. Woods (1974)
"Control Concepts in a Speech Understanding System," BBN Report No. 2703, Bolt Beranek and Newman Inc., Cambridge MA.

[15] Vicens, P. (1969)
"Aspects of Speech Recognition by Computer," Report CS-127, Ph.D. Thesis, Computer Science Department, Stanford University.

[16] Woods, W.A., M. Bates, G. Brown, B. Bruce, C. Cook, L. Gould, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue (1975)
"Speech Understanding Systems, Quarterly Technical Progress Report 4, 31 July 1975 to 29 October 1975," BBN Report No. 3188, Bolt Beranek and Newman Inc., Cambridge MA.

[17] Woods, W.A., M. Bates, G. Brown, B. Bruce, C. Cook, L. Gould, J. Klovstad, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue (1976a)
"Speech Understanding Systems, Quarterly Technical Progress Report 5, 30 October 1975 to 31 January 1976," BBN Report No. 3240, Bolt Beranek and Newman Inc., Cambridge MA.

[18] Woods, W.A., M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue (1976b)
"Speech Understanding Systems, Quarterly Technical Progress Report No. 6, 1 February 1976 to 30 April 1976," Report No. 3303, Bolt Beranek and Newman Inc., Cambridge MA.

[19] Woods, W.A., M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, R. Schwartz, J. Wolf (1976c)
"Speech Understanding Systems, Quarterly Technical Progress Report No. 7, 1 May 1976 to 31 July 1976, Report No. 3359, Bolt Beranek and Newman Inc., Cambridge MA.

[20] Woods, W.A. (1974)
"Motivation and Overview of BBN SPEECHLIS: an Experimental Prototype for Speech Understanding Research," Proc. IEEE Symposium on Speech Recognition, CMU, pp. 2-10.

[21] Woods, W.A. (1975)
"A Phonological Rule System for Dictionary Expansion," in Woods et al., (1975), pp. 68-94.

[22] Woods, W.A. (1976)
"Shortfall Scoring Strategies for Speech Understanding Control", in Woods et al., (1976b).

[23] Woods, W.A. and J.I. Makhoul (1973)
"Mechanical Inference Problems in Continuous Speech Understanding," Proceedings of the Third International Joint Conference on Artificial Intelligence, pp. 200-207.

[24] Woods, W.A. and V. Zue (1976)
"Dictionary Expansion via Phonological Rules for a Speech Understanding System," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, April 12-14, Philadelphia, pp. 561-564.

Appendix 1 - Annotated Phonological Rules

Define phonemes as feature bundles:

```
[P      (PHONEME (CONS OBST LAB ANT) )]
[B      (PHONEME (CONS VCD OBST LAB ANT) )]
[F      (PHONEME (CONS OBST LAB ANT CONT) )]
[V      (PHONEME (CONS VCD OBST LAB ANT CONT) )]
[W      (PHONEME (CONS HIGH SONOR BACK GLIDE ROUND VCD LAB) )]
[M      (PHONEME (CONS NASAL SONOR VCD OBST LAB ANT) )]
[T      (PHONEME (CONS COR OBST ANT) )]
[D      (PHONEME (CONS COR VCD OBST ANT) )]
[TH     (PHONEME (CONS DIST COR OBST ANT CONT) )]
[DH     (PHONEME (CONS DIST COR VCD OBST ANT CONT) )]
[N      (PHONEME (CONS COR NASAL SONOR VCD OBST ANT) )]
[S      (PHONEME (CONS STRID COR OBST ANT CONT) )]
[Z      (PHONEME (CONS STRID COR VCD OBST ANT CONT) )]
[SH     (PHONEME (CONS STRID COR HIGH OBST ANT CONT) )]
[CH     (PHONEME (CONS STRID COR HIGH OBST ANT) )]
[ZH     (PHONEME (CONS STRID COR HIGH VCD OBST ANT CONT) )]
[JH     (PHONEME (CONS STRID COR HIGH VCD OBST ANT) )]
[DX     (PHONEME (CONS COR SONOR VCD ANT) )]
[NX     (PHONEME (CONS NASAL HIGH SONOR BACK VCD OBST) )]
[NY     (PHONEME (CONS NASAL HIGH SONOR VCD OBST) )]
[KY     (PHONEME (CONS HIGH OBST) )]
[GY     (PHONEME (CONS HIGH VCD OBST) )]
[K      (PHONEME (CONS HIGH BACK OBST) )]
[G      (PHONEME (CONS HIGH BACK VCD OBST) )]
[Y      (PHONEME (CONS HIGH SONOR GLIDE VCD ANT) )]
[L      (PHONEME (CONS LAT COR SONOR VCD ANT CONT) )]
[R      (PHONEME (CONS RETRO COR SONOR VCD ANT CONT) )]
[-L     (PHONEME (CONS LAT COR HIGH SONOR BACK VCD CONT) )]
[-R     (PHONEME (CONS RETRO COR HIGH SONOR BACK VCD CONT) )]
[HH     (PHONEME (LOW ASP GLIDE) )]
[Q      (PHONEME (LOW GLIDE) )]
[HW     (PHONEME (HIGH BACK GLIDE ROUND LAB) )]
[URP    (PHONEME (CONS UNREL OBST LAB ANT) )]
[URT    (PHONEME (CONS UNREL COR OBST ANT) )]
```

```
[URK    (PHONEME (CONS UNREL HIGH BACK OBST) )]
[URB    (PHONEME (CONS UNREL VCD OBST LAB ANT) )]
[URD    (PHONEME (CONS UNREL COR VCD OBST ANT) )]
[URG    (PHONEME (CONS UNREL HIGH BACK VCD OBST) )]
[IY     (PHONEME (VOWEL ATR HIGH SONOR VCD ANT) )]
[IH     (PHONEME (VOWEL HIGH SONOR VCD ANT) )]
[EY     (PHONEME (VOWEL ATR SONOR VCD ANT) )]
[EH     (PHONEME (VOWEL SONOR VCD ANT) )]
[EA     (PHONEME (VOWEL ATR LOW SONOR VCD ANT) )]
[AE     (PHONEME (VOWEL LOW SONOR VCD ANT) )]
[IX     (PHONEME (REDUCED VOWEL HIGH SONOR VCD ANT) )]
[AX     (PHONEME (REDUCED VOWEL SONOR VCD) )]
[UW     (PHONEME (VOWEL ATR HIGH SONOR BACK ROUND VCD) )]
[UH     (PHONEME (VOWEL HIGH SONOR BACK ROUND VCD) )]
[OW     (PHONEME (VOWEL SONOR BACK ROUND VCD) )]
[AO     (PHONEME (VOWEL LOW SONOR BACK ROUND VCD) )]
[AW     (PHONEME (DIPHTHONG VOWEL LOW SONOR BACK ROUND VCD) )]
[AY     (PHONEME (DIPHTHONG VOWEL LOW SONOR BACK VCD) )]
[OY     (PHONEME (DIPHTHONG VOWEL SONOR BACK ROUND VCD) )]
[AA     (PHONEME (VOWEL LOW SONOR BACK VCD) )]
[AH     (PHONEME (VOWEL SONOR BACK VCD) )]
[ER     (PHONEME (VOWEL RETRO COR HIGH SONOR BACK VCD) )]
[EL     (PHONEME (VOWEL LAT COR HIGH SONOR BACK VCD) )]
[EN     (PHONEME (VOWEL COR NASAL SONOR VCD ANT) )]
[EM     (PHONEME (VOWEL NASAL SONOR VCD LAB ANT) )]
[AXR    (PHONEME (VOWEL RETRO COR SONOR VCD) )]
[HIY    (PHONEME (VOWEL ATR HIGH SONOR ANT) )]
[HIH    (PHONEME (VOWEL HIGH SONOR ANT) )]
[UIX    (PHONEME (REDUCED VOWEL HIGH SONOR ANT) )]
[HUW    (PHONEME (VOWEL ATR HIGH SONOR BACK ROUND) )]
[HUH    (PHONEME (VOWEL HIGH SONOR BACK) )]
[UAX    (PHONEME (REDUCED VOWEL SONOR) )]
[TR     (PHONEME (CONS VCLESS ASP RETRO COR OBST ANT) )]
[DR     (PHONEME (CONS RETRO COR VCD OBST ANT) )]
[SR     (PHONEME (CONS VCLESS RETRO STRID COR OBST ANT CONT) )]
[ZR     (PHONEME (CONS RETRO STRID COR VCD OBST ANT CONT) )]
[UY     (PHONEME (VOWEL ATR HIGH SONOR ROUND VCD ANT) )]
```

Rule tree used for phonological rule expansion of the dictionary:

```
ZUELIST ((OPT Z3 .7 .5)
         (OPT Z3A .7 .5)
         (OPT Z4 .7 .5)
         (OPT Z4A .7 .5)
         (OPT Z5 .7 .5)
         (OPT Z5A .7 .5)
         (OPT Z6 .7 .5)
         (OPT Z6A .7 .5)
         (OPT Z7 .7 .5)
         (OPT Z8 .7 .5)
         (OPT Z8A .7 .5)
         (OPT Z1 .7 .5)
         (OPT Z2 .7 .5)
         (OPT FR1)
         R46 R16 R16.1 (OPT Z16 1.0 1.0)
         (OPT Z56 1.0 .2)
         (OPT R17 .8 1.0)
         (OPT Z18 .8 1.0)
         (OPT Z19 .8 1.0)
         (OPT Z58)
         (OPT R10 .8 1.0)
         R10B
         (OPT Z22.4)
         (OPT Z22.1)
         (OPT Z22.2)
         (OPT Z22.3 .1 1.0)
         (IF (OPT R26 1.0 .6)
             THEN R27)
         (OPT R32 1.0 1.0)
         (ELSE (OPT R32B))
         (OPT Z60)
         (IF (OPT R50 1.0 .5)
             THEN R55)
         (OPT Z13A 1.0 .2)
         (ELSE (OPT Z13B 1.0 .2)
               (ELSE (OPT Z13C 1.0 .2)
                     (ELSE (OPT Z13D 1.0 .2))))
         (OPT Z13E 1.0 .5)
         (OPT Z57)
         (OPT R30.Z .5 1.0))
```

Rule tree used for APR rule expansion of the dictionary:

```
APRLIST ((OPT APRND .5 1.0)
         (OPT APRND2 .5 1.0)
         APR1 APR1A APR2 APR2A APR3 (IF APR4 THEN R55)
         APR15 APR16 APR10 APR11 APR12 APR13 (OPT APR14)
         APR17 APR17A APR17B))
```

Listing of the rules:

```
** Rule R-EST.A through R-ING are rules used for deriving regular
** inflections from the uninflected base forms, depending on the
** features of the last segment.

[R-EST.A
      (RULE ((* S (EITHER P OR T OR K) -EST)
             (* (0 *)
                *I
                (: IX !- S T))) )]
[R-EST.B
      (RULE ((* NX -EST)
             (* *I (: * IX !- S T))) )]
[R-EST.C
      (RULE ((* (+ CONS)
                -EST)
             (* (0 *)
                (: IX !- S T))) )]
[R-EST
      (RULE (-EST (: * IX !- S T)) )]
[R-ER.A
      (RULE ((* S (EITHER P OR T OR K) -ER)
             (* (0 *)
                *I
                (: AXR !-))) )]
[R-ER.B
      (RULE ((* NX -ER)
             (* *I (: * AXR !-))) )]
[R-ER.C
      (RULE ((* (+ CONS) -ER)
             (* (0 *)
                (: AXR !-))) )]
[R-ER
      (RULE (-ER (: * AXR !-)) )]
[R-ED1.A
      (RULE ( * S T -ED)
             (* (0 *)
                *I
                (: IX !- D))) )]
[R-ED1.B
      (RULE ((* (EITHER T OR D) -ED)
             (* (0 *)
                (: IX !- D))) )]
[R-ED2
      (RULE (-ED T / (+ CONS - VCD)  --) )]
[R-ED3
      (RULE (-ED D) )]
[R-S1
      (RULE ((* (EITHER S OR Z OR SH OR CH OR ZH OR JH)
                -S)
             (* (0 *)
                (: IX !- Z))) )]
[R-S2
      (RULE (-S S / (+ CONS - VCD) --) )]
[R-S3
      (RULE (-S Z) )]
[R-ING.A
      (RULE ((* S (EITHER P OR T OR K) -ING)
             (* (0 *)
                *I
                (: IX !- NX))) )]
[R-ING.B
      (RULE ((* NX -ING)
             (* *I (: * IX !- NX))) )]
[R-ING.C
      (RULE ((* (+ CONS) -ING)
             (* (0 *)
                (: IX !- NX))) )]
[R-ING
      (RULE (-ING (: * IX !- NX)) )]
```

```
** Rules Z3 through Z2 are used for vowel reduction.  Note
** that certain vowels are not allowed to be reduced if they appear
** as the last segment of a word.

[Z3
     (RULE (IY IH / -- !0 (EITHER (* (# 1 3 (+ CONS))
                                     #)
                                  OR
                                  (* * (+ CONS)))
               $
               (+ VOWEL)) )]
[Z3A
     (RULE (IY IH / # (# 0 3 (+ CONS))
               -- !0 (# 0 2 (+ CONS))
               #)]
[Z4
     (RULE (EY EH / -- !0 (EITHER (* (# 1 3 (+ CONS))
                                     #)
                                  OR
                                  (* * (+ CONS)))
               $
               (+ VOWEL)) )]
[Z4A
     (RULE (EY EH / # (# 0 3 (+ CONS))
               -- !0 (# 0 2 (+ CONS))
               #) )]
[Z5
     (RULE (UW UH / -- !0 (EITHER (* (# 1 3 (+ CONS))
                                     #)
                                  OR
                                  (* * (+ CONS)))
               $
               (+ VOWEL)) )]
[Z5A
     (RULE (UW UH / # (# 0 3 (+ CONS))
               -- !0 (# 0 2 (+ CONS))
               #) )]
[Z6
     (RULE (OW AH / -- !0 (EITHER (* (# 1 3 (+ CONS))
                                     #)
                                  OR
                                  (* * (+ CONS)))
               $
               (+ VOWEL)) )]
[Z6A
     (RULE (OW AH / # (# 0 3 (+ CONS))
               -- !0 (# 0 2 (+ CONS))
               #) )]
[Z7
     (RULE ((* ER !0)
            (* AXR !-)) )]
[Z8
     (RULE (AW AA / -- !0 (EITHER (* (# 1 3 (+ CONS))
                                     #)
                                  OR
                                  (* * (+ CONS)))
               $
               (+ VOWEL)) )]
[Z8A
     (RULE (AW AA / # (# 0 3 (+ CONS))
               -- !0 (# 0 2 (+ CONS))
               #)]
[Z1
     (RULE ((* IH !0)
            (* IX !-))]
[Z2
     (RULE ((* (EITHER EH OR AE OR AA OR AO OR AH OR UH)
               !0)
            (* AX !-)) )]
```

```
** Front rounding rule, change UW to a front rounded vowel UY.
** e.g. "New York"   N UW Y AO R K => N UY Y AO R K


[FR1
      (RULE (UW UY / (EITHER (+ COR - LAT - RETRO)
                             OR Y)
                   --
                   (EITHER !2 OR !1 OR !0)
                   (OPT *)
                   (EITHER (+ COR - LAT - RETRO)
                           OR Y))]


** This rule is used to clean up after the vowel reduction rules.
** It deletes one of the schwas if ttwo appear in sequence.


[R46
      (RULE ((* AX !-)
             0 / AX !- --) )]


** Syllabification of R.
** e.g. "for"   F AX R => F AXR


[R16
      (RULE ((* AX !- R)
             (* AXR !- 0)))]


** Syllabification of R across syllable boundary.
** e.g. "arrange"   AX R EY N JH => AXR EY N JH

[R16.1
      (RULE ((* AX !- * R)
             (* AXR !- * 0)))]


** Desyllabify AXR when it appears between vowels
** e.g. "covering"   K AH V AXR IX NX => K AH V AX R IX NX

[Z16
      (RULE ((* AXR !- *)
             (* AX !- (: * R))
             / -- (+ VOWEL)) )]


** Transitional glide insertion for W
** e.g. "going"   G OW IX NX => G OW W IX NX

[Z56
      (RULE (0 W / (EITHER OW OR AW OR UW)
                   (EITHER !2 OR !1 OR !0)
                   * -- (+ VOWEL + ANT))]


** Syllabification of L.
** e.g. "travel"   T R AE V AX L => T R AE V EL

[R17
      (RULE ((* (EITHER AX OR IX)
                !- L)
             (* EL !- 0)
             / -- (EITHER (+ CONS)
                          OR #)) )]
```

```
** Syllabification of N.
** e.g. "London"   L AH N D AX N => L AH N D EN

[Z18
      (RULE ((* (EITHER AX OR IX)
                !- N)
             (* EN !- 0)
             /
             (+ CONS)
             --
             (EITHER (+ CONS)
                     OR #)) )]


** Syllabification of M.
** e.g. "item"   AY T AX M => AY T EM

[Z19
      (RULE ((* (EITHER AX OR IX)
                !- M)
             (* EM !- 0)
             /
             (+ CONS)
             --
             (EITHER (+ CONS)
                     OR #)) )]


** Glottalization of T.
** e.g. "gotten"   G AA T EN => G AA Q EN

[Z58
      (RULE (T Q / !2 * -- (EITHER EN OR EL))]


** The next two rules are for homorganic stop insertion.
** e.g. "since"   S IH N S => S IH N T S

[R10
      (RULE (0 (+ OBST + CONS + ANT - CONT (X)
                  LAB
                  (Y)
                  COR
                  (Z)
                  VCD)
               /
               (+ NASAL X LAB Y COR)
               (OPT !-)
               --
               (OPT *)
               (+ CONT - SONOR Z VCD (X)
                  LAB
                  (Y)
                  COR))]
[R10B
      (RULE ((* T * SH)
             (* 0 * CH)) )]


** Palatalization rule that inserts Y before UW.
** e.g. "assume"   AX S UW M => AX S Y UW M

[Z22.4
      (RULE (0 Y / (EITHER S OR N OR JH)
               -- UW) )]
```

```
** Palatalization rule for fricatives
** e.g. "misuse"   M IH S Y UW Z => M IH SH Y UW Z

[Z22.1
      (RULE ((EITHER S OR Z)
             (+ HIGH)
             /
             (+ VOWEL + HIGH)
             --
             (OPT *)
             (+ HIGH + ANT - CONS)
             (OPT *)
             (+ VOWEL)) )]


** Palatalization rule for Y-IY-SCHWA sequence.
** e.g. "California"   K AE L F AO R N IY AX => K AE L F AO R N Y AX

[Z22.2
      (RULE ((* IY (EITHER !1 OR !0)
                *)
             (* Y 0 0)
             /
             (+ CONS + COR)
             (OPT *)
             --
             (EITHER IX OR AX)) )]


** Palatalization rule for stops
** e.g. "Tuesday"   T Y UW Z D EY => CH Y UW Z D EY

[Z22.3
      (RULE ((EITHER T OR D)
             (+ HIGH + STRID)
             / -- (OPT *)
             Y
             (OPT *)
             (+ VOWEL)
             (EITHER !2 OR !1 OR !0))]


** The next two rules are for HW unrounding.
** e.g. "which"   HH W IH CH => W IH CH

[R26
      (RULE ((* HH W)
             (* 0 HW)) )]
[R27
      (RULE (HW W) )]


** RUH reduction rule (Case A): RUH is preceded by at least one
** consonant and followed by a vowel.
** e.g. "propose"   P R AX P OW Z => P AXR P OW Z

[R32
      (RULE ((* R (EITHER AX OR IX)
                !-)
             (* AXR 0 !-)
             /
             (+ CONS)
             -- $ (+ VOWEL)
             (EITHER !2 OR !1))]
```

```
** RUH reduction rule (Case B): RUH is following a stressed vowel
** and zero or more consonants (but not an NX).
** e.g. "hundred"   HH AH N D R IX D => HH AH N D AXR D

[R32B
      (RULE ((* R (EITHER AX OR IX)
                  !-)
             (* AXR 0 !-)
             /
             (+ VOWEL)
             (EITHER !2 OR !1)
             (# 0 2 (+ CONS))
             (OPT *)
             (# 0 2 (+ CONS))
             --
             (EITHER (- NASAL)  OR  (- BACK)  OR
                     (- SONOR)  OR  (- HIGH)  OR
                     (- VCD)  OR  (- OBST))])


** Stop deletion rule
** e.g. "arrange"   AX R EY N JH => AX R EY N ZH

[Z60
      (RULE (JH ZH / N -- #)]


** Schwa deletion rule; applies to a schwa when it is preceded by a
** stressed syllable and followed by a unstressed syllable.  The
** resulting extra syllable boundary is removed by R55.
** e.g. "original"   AO R IH G IX N EL => AO R IH G N EL

[R50
      (RULE ((* (EITHER AX OR IX)
                !-)
             (* 0 0)
             /
             (+ VOWEL)
             !2
             (# 0 2 (+ CONS))
             *
             (# 0 2 (+ CONS - GLIDE))
             -- * (EITHER N OR M OR R OR L)
             (+ VOWEL)
             (EITHER !0 OR !-))]


** Rules Z13A-E are various forms of the flapping rule for dental stops.
** e.g. "enter"   EH N T AXR => EH N DX AXR

[Z13A
      (RULE ((EITHER T OR D)
             DX / (+ VOWEL)
             !2
             (OPT R)
             (OPT *)
             --
             (OPT *)
             (+ VOWEL)
             (EITHER !0 OR !-))]
      (RULE ((EITHER T OR D)
             DX / (+ VOWEL)
             !1
             (OPT R)
             (OPT *)
             --
             (OPT *)
             (+ VOWEL)
             (EITHER !0 OR !-)) )]
```

```
[Z13C
      (RULE ((EITHER T OR D)
             DX / (+ VOWEL)
             !0
             (OPT R)
             (OPT *)
             --
             (OPT *)
             (+ VOWEL)
             (EITHER !0 OR !-)) )]

[Z13D
      (RULE ((EITHER T OR D)
             DX / (+ VOWEL)
             !-
             (OPT R)
             (OPT *)
             --
             (OPT *)
             (+ VOWEL)
             !-) )]
[Z13E
      (RULE (T DX / (+ VOWEL)
               (EITHER !2 OR !1 OR !0 OR !-)
               N
               (OPT *)
               --
               (OPT *)
               (+ VOWEL)
               (EITHER !0 OR !-))]


** Flapping rule for N.
** e.g. "any"    EH N IY => EH TX IY

[Z57
      (RULE (N TX / !2 * -- IY !0 (OPT *)
               (EITHER (+ CONS) OR #))]


** schwa devoicing rule; applies to schwa between two voiceless stops.
** e.g. "multiply"    M AH L T UIX P L AY => M AH L T P L AY

[R30.Z
      (RULE ((EITHER AX OR IX)
             (- VCD)
             /
             (EITHER (- VCD + CONS - CONT + OBST)  OR JH)
             (OPT *)
             -- !- (OPT *)
             (- VCD + CONS - CONT + OBST)
             $
             (+ VOWEL))]

[R55
      (RULE (* 0 / -- (# 0 2 (+ CONS))  *)]


** nasal deletion rule; if APR could detect nasalized vowels, this would
** be replaced by two legitimate phonological rules, the first of
** which would nasalize the vowel optionally and the second of which wou
**ld delete
** the nasal optionally after a nasalized vowel.
** e.g. "spent"    S P EH N T => S P EH T

[APRND
      (RULE (N 0 / !2 -- (OPT *)
               (+ CONS))]


** Nasal deletion rule for i.
** e.g. "assumption"    AX S AH M P SH EN => AX S AH P SH EN

[APRND2
      (RULE (M 0 / !2 -- (OPT *)
               (+ CONS + LAB))]
```

```
** rules APR1 through APR3 are allophonic rules for vowels.
** e.g. "trip"   T R IH P => T R RIH P

[APR1
      (RULE (IH RIH / R -- !2)]
[APR1A
      (RULE (IH RIH / -- !2 R) )]
[APR2
      (RULE (IH LIH / L -- !2)]
[APR2A
      (RULE (IH LIH / -- !2 L)]
[APR3
      (RULE (IX INX / -- !- NX)]

** voiceless schwa deletion (The APR currently does not detect voiceless
** schwa)
** e.g. multiply

[APR4
      (RULE ((* (EITHER UIX OR UAX)
                !-)
             (* 0 0))]

** rules APR10 through APR13 introduces allophones for T.

[APR10
      (RULE (T TG / -- (EITHER R OR W OR L OR Y))]
[APR11
      (RULE (T ST / (+ STRID)
                --
                (+ VOWEL))]
[APR12
      (RULE (T TV / -- (+ VOWEL))]
[APR13
      (RULE (T TS / -- (EITHER S OR Z OR SH OR ZH))]

** allophone for CH

[APR14
      (RULE (CH (: TS TCH))]

** APR rule for converting AX to IX based on context.

[APR15
      (RULE (AX IX /
                (EITHER T OR D OR N OR K OR G OR NX OR S OR Z OR CH OR J
**H)
                -- !- (OPT *)
                (EITHER T OR D OR N OR K OR G OR NX OR S OR Z OR CH OR J
**H))]

** allophone for back K

[APR16
      (RULE (K KA / -- (EITHER (+ BACK + VOWEL)
                               OR AX OR AXR OR R OR W OR L))]

** rules APR17 through APR17B create allophone for nasals.

[APR17
      (RULE (N YN /
                (EITHER IY OR EY OR R OR ER OR AXR)
                (EITHER !2 OR !1 OR !0 OR !-)
                (OPT *)  --)]
[APR17A
      (RULE (M YM /
                (EITHER IY OR EY OR R OR ER OR AXR)
                (EITHER !2 OR !1 OR !0 OR !-)
                (OPT *)  --) )]
[APR17B
      (RULE (NX N / AO (EITHER !2 OR !1 OR !0 OR !-) --)]
```

Program to Generate Across Word Phonological Rules

```
static { OutJfn := nil

        // Stress 2 vowels
  S2Vwl :=  table 19,
        "IY!2", "IH!2", "EH!2", "AE!2", "AA!2", "AH!2", "AO!2",
        "UH!2", "UW!2", "ER!2", "EY!2", "OW!2", "AW!2", "AY!2",
        "OY!2", "UY!2", "EA!2", "LIH!2", "RIH!2"

        // Stressed vowels
  SVwl :=  table 38,
        "IY!2", "IH!2", "EH!2", "AE!2", "AA!2", "AH!2", "AO!2",
        "UH!2", "UW!2", "ER!2", "EY!2", "OW!2", "AW!2", "AY!2",
        "OY!2", "UY!2", "EA!2", "LIH!2","RIH!2","IY!1", "IH!1",
        "EH!1", "AE!1", "AA!1", "AH!1", "AO!1", "UH!1", "UW!1",
        "ER!1", "EY!1", "OW!1", "AW!1", "AY!1", "OY!1", "UY!1",
        "EA!1", "LIH!1", "RIH!1"

        // High back vowels for W-deletion
  HBVwl :=  table 3,
        "AW!2", "OW!2", "UW!2"

        // Weak vowels for Flapped T rule
  WkVwl :=  table 4,
        "IH!1", "IH!0", "AX!-", "IX!-"

        // List of all vowels
  Vwl :=    table 66,
        "IY!0", "IY!1", "IY!2", "IH!0", "IH!1", "IH!2", "EH!0",
        "EH!1", "EH!2", "AE!0", "AE!1", "AE!2", "AA!0", "AA!1",
        "AA!2", "AH!0", "AH!1", "AH!2", "AO!0", "AO!1", "AO!2",
        "UH!0", "UH!1", "UH!2", "UW!0", "UW!1", "UW!2", "AX!-",
        "ER!0", "ER!1", "ER!2", "EL!-", "EY!0", "EY!1", "EY!2",
        "OW!0", "OW!1", "OW!2", "AW!0", "AW!1", "AW!2", "AY!0",
        "AY!1", "AY!2", "OY!0", "OY!1", "O    "RIH!1","RIH!2","INX!-"

        // These stops can be changed to URPlosives
  Stp :=    table 6,
        "P", "T", "K", "B", "D", "G"
  URStp :=  table 6,
        "URP", "URT", "URK", "URB", "URD", "URG"
        // Right context for UR stop rule
  Cnst1 :=  table 14,
        "P", "TG", "TV", "TS", "K",  "B",  "D",
        "G", "F",  "TH", "JH", "CH", "DH", "-"

        // Pre UY context
  PreUY :=  table 6,
        "TV", "ST", "D", "HH", "N", "Y"
        // post UY context
  CrCon :=  table 8,
        "D", "TV", "TG", "TS", "S", "N", "TH", "Y"

        // V deletion
  Cnst2 :=  table 7,
        "M", "B", "D", "DH", "F", "P", "HH"

        // DH deletion
  Cnst3 :=  table 11,
        "N", "Z", "CH", "TCH", "T", "JH", "ZH", "TH",
        "M", "S", "-"

        // D Deletion
  Cnst4 :=  table 10,
        "DH", "TG", "TV", "TS", "CH", "S", "L", "W"
```

```
        // T Deletion
  UVFric := table 4,
          "S", "F", "DH", "TH"

        // Geminate reduction
  GCnst :=  table 12,
          "P", "K", "B", "D", "G", "F",
          "S", "Z", "SH","TH","M", "N"

        // HH Deletion
  Strd1 :=  table 8,
          "S", "SH", "CH", "TCH", "DH", "P", "T", "K"

    // context for creating allophones of T
        // ST allophone
  Strd2 :=  table 1,
          "S"
        // TV allophone
  NStrd :=  table 97,
          "IY!0", "IY!1", "IY!2", "IH!0", "IH!1", "IH!2",
          "EH!0", "EH!1", "EH!2", "AE!0", "AE!1", "AE!2",
          "AA!0", "AA!1", "AA!2", "AH!0", "AH!1", "AH!2",
          "AO!0", "AO!1", "AO!2", "UH!0", "UH!1", "UH!2",
          "UW!0", "UW!1", "UW!2", "AX!-", "ER!0", "ER!1",
          "ER!2", "EL!-", "EY!0", "EY!1", "EY!2", "OW!0",
          "OW!1", "OW!2", "AW!0", "AW!1", "AW!2", "AY!0",
          "AY!1", "AY!2", "OY!0", "OY!1", "OY!2", "L",
          "W",    "R",    "Y",    "N",    "M",    "NX",
          "P",    "K",    "B",    "D",    "G",    "F",
          "TH",   "V",    "DH",   "HH",   "DX",   "EN!-",
          "EM!-", "ENX",  "URP",  "URT",  "URK",  "URB",
          "URD",  "URG",  "IX!-", "AXR!-","UY!0", "UY!1",
          "UY!2", "EA!0", "EA!1", "EA!2", "UAX!-","TX",
          "Q",    "-",    "UIX!-","ST",   "TV",   "TG",
          "TS",   "LIH!1","LIH!2","RIH!0","RIH!1","RIH!2",
          "INX!-"
        // TS allophone
  Strd3 :=  table 3,
          "S", "SH","Z"
        // TG allophone
  Gld :=    table 4,
          "R", "W", "L", "Y"

        // left context for YN and YM allophones
  NasB :=  table 11,
          "R",
          "AXR!-",
          "ER!0","ER!1","ER!2",
          "IY!0","IY!1","IY!2",
          "EY!0","EY!1","EY!2"
 }

let WtStrs(String,nil repname 20) be { let Strings := lv String
  WriteS(OutJfn,String)
  for i := 1 to NumbArgs()-1 do
  { BOUT(OutJfn,$*s) ; WriteS(OutJfn,Strings!i)
   }
  WriteS(OutJfn,"*c*l")
  return
 }

and Start() be { WriteS("Rules Generated are to go to")
  OutJfn := CreateOutput(1)

// unreleased stops before obstruents
  for i := 1 to Stp!0 do
  { for j := 1 to Cnst1!0 do
    { WtStrs(Stp!i,"#",Cnst1!j,"=>",URStp!i,Cnst1!j)
     }
   }
```

```
// V - deletion (mostly before labials)
  for i := 1 to Cnst2|0 do
  { WtStrs("V #",Cnst2|i,"=>",Cnst2|i)
   } // This is an APR rule for when it is called obstruent
  WtStrs("V # M => V")

// DH - deletion (after dentals, palatals, M, -
  for i := 1 to Cnst3|0 do
  { WtStrs(Cnst3|i,"# DH =>",Cnst3|i)
   }

// HH - deletion (after frication or aspiration)  //    Is  really  a  rule
which applies to HAVE,HAS,HAD etc.
  for i := 1 to Strd1|0 do
  { WtStrs(Strd1|i,"# HH AE!2 =>",Strd1|i,"AE!2")
   }

// D - deletion (following N, preceding many dental consonants)
  for i := 1 to Cnst4|0 do
  { WtStrs("N D #",Cnst4|i,"=> N",Cnst4|i)
   }

// W - deletion (following HBVwl)
  for i := 1 to HBVwl|0 do
  { WtStrs(HBVwl|i,"# W =>",HBVwl|i)
   }

// T - deletion (between S and UVFric
  for i := 1 to UVFric|0 do
  { WtStrs("S T #",UVFric|i,"=> S",UVFric|i)
   }

// T - deletion rule (special case for S)
  WtStrs("S T # S => S")

// Flapped T rules - right now, following a Vwl, preceding a Vwl.
  for i := 1 to Vwl|0 do
  { for j := 1 to Vwl|0 do
        // Word ending T or D - allow for any vowel.
    { WtStrs(Vwl|i,"T #",Vwl|j,"=>",Vwl|i,"DX",Vwl|j)
      WtStrs(Vwl|i,"D #",Vwl|j,"=>",Vwl|i,"DX",Vwl|j)
     }
   }

  for j := 1 to WkVwl|0 do
        // Word initial T - allow following S2Vwl and preceding WkVwl.
  { for i := 1 to S2Vwl|0 do
      WtStrs(S2Vwl|i,"# TV",WkVwl|j,"=>",S2Vwl|i,"DX",WkVwl|j)
        // for word initial T or D preceded by R, require
        // preceded by SVwl and followed by WkVwl
    for i := 1 to SVwl|0 do
    { WtStrs(SVwl|i,"R # TV",WkVwl|j,"=>",SVwl|i,"R DX",WkVwl|j)
      WtStrs(SVwl|i,"R # D",WkVwl|j,"=>",SVwl|i,"R DX",WkVwl|j)
     }
   }

// Palatization rules.
  WtStrs("S # Y => SH")
  WtStrs("S # SH => SH")
  WtStrs("Z # SH => SH")
  WtStrs("D # Y => JH")

// Geminate reduction
  for i := 1 to GCnst|0 do
  { WtStrs(GCnst|i,"#",GCnst|i,"=>",GCnst|i)
   } // specific geminate reduction rules for T allophones
  WtStrs("T # TV => TV")
  WtStrs("T # TG => TG")
  WtStrs("T # TS => TS")
```

```
// These are deletion due to same place of articulation
  WtStrs("T # D => D")
  WtStrs("D # TV => TV")
  WtStrs("D # TG => TG")
  WtStrs("D # TS => TS")
  WtStrs("P # B => B")
  WtStrs("B # P => P")
  WtStrs("K # G => G")
  WtStrs("G # K => K")


// these are APR geminate reduction rules.
  WtStrs("Z # S => S")
  WtStrs("SH # S => S")
  WtStrs("CH # SH => CH")
  WtStrs("N # M => N")
  WtStrs("M # N => M")

// The following rules allow the 5 allophones of T.


//      preceded by StFRIC
  for i := 1 to Strd2|0 do
  { for j := 1 to Vwl|0 do
    { WtStrs(Strd2|i,"T #",Vwl|j,"=>",Strd2|i,"ST",Vwl|j)
    }
  } //      Prevocalic T
  for i := 1 to NStrd|0 do
  { for j := 1 to Vwl|0 do
    { WtStrs(NStrd|i,"T #",Vwl|j,"=>",NStrd|i,"TV",Vwl|j)
    }
  } //      Followed by StFRIC
  for i := 1 to Strd3|0 do
  { WtStrs("T #",Strd3|i,"=> TS",Strd3|i)
  }

//  Allows  Glottal  stops  at phrase beginnings and //between vowels and a
stressed vowel.
  for i := 1 to S2Vwl|0 do
  { WtStrs("- #",S2Vwl|i,"=> - Q",S2Vwl|i)
  }
  for i := 1 to Vwl|0 do
  { for j := 1 to S2Vwl|0 do
    { WtStrs(Vwl|i,"#",S2Vwl|j,"=>",Vwl|i,"Q",S2Vwl|j)
    }
  }

// UW  can  become  fronted  when  preceded  and  followed  by  coronal  //
consonants.
  for i := 1 to CrCon|0 do
    for j := 1 to PreUY|0 do
    { WtStrs(PreUY|j,"UW!2 #",CrCon|i,"=>",PreUY|j,"UY!2",CrCon|i)
    }


// This makes betas of the N and M
  for i := 1 to NasB|0 do
  { WtStrs(NasB|i,"# N =>",NasB|i,"YN")
    WtStrs(NasB|i,"# M =>",NasB|i,"YM")
  }

  EndWrite(OutJfn)
  finish
 }
```

Appendix 2

Formats and Examples of Dictionary Files

In this appendix, we give sample fragments of the various files produced during dictionary expansion (see Sec. C). With the exception of the VERDICT and DICTTEXT files, all of them are LISP symbolic files with appropriate dictionary information recorded on the property list of each entry. The format for property lists on such files is

```
[word
      (property1 value1
       property2 value2
       .
       .
       .
       propertyn valuen ) ]
```

TRAVELDICT.FRAGMENT is a fragment of the original dictionary file before expansion. Here, properties include the syntactic categories of which the word is a member (e.g., SPECIAL, AUX, V, PREP, N, SPONSOR, ADJ) as well as other kinds of information. For example, the PHONEMES property has as its value the phonetic pronunciation or pronunciations associated with the word. For each syntactic category, the value of the property indicates how the word forms its inflections orthographically. For example, an -S value for the N property indicates that, as a noun, the word forms its plural by adding -S. For verbs (V) only third person singular and past inflectional affixes are indicated, as all regular verbs form their present progressive by adding -ing. The value * after a syntactic category indicates that a word undergoes the default inflection for that category (no inflection for most categories, but regular -er, -est inflection for adjectives). Inflectional codes not shown in the fragment are S-ED, ES-ED, S-D, IRR, and MASS.

The value of the PHONEMES property is either a single ARPAbet "spelling" for the word, or a list of the form ((OR sp1 sp2 ... spn)) where each spi is an alternative spelling. The general format for a phonetic spelling is:

((p f1 f2 ... fn) ph1 ph2 ... phn)

where p is a pronunciation likelihood for the spelling (a number indicating the likelihood of this particular pronunciation of the word), the fi's are pronunciation features associated with this pronunciation (e.g., REDUCED is a feature of some pronunciations of some function words), and the phi's are the phonetic elements which make up the particular pronunciations. These elements include "ARPAbet" phonemes, the symbol * (indicating syllable boundaries), and the symbols !2, !1, !0, and !- (indicating decreasing levels of stress). Both the pronunciation likelihood and the pronunciation features are optional and may be omitted.

INFLECTED.FRAGMENT is a fragment of the file produced as a byproduct of the initial inflectional expansion. Its format is similar to that of the original dictionary, except for the addition of the properties TREE and WORDINDEX to each entry and the creation of new entries for regular inflected forms. The WORDINDEX property assigns a numerical index to each dictionary entry to be used for referring to words by number in interfork communication and internal arrays. The TREE property contains pronunciations in the form required for input to the rule expansion program. This format is similar to that of the PHONEMES property, except that there is always a pronunciation likelihood associated with each pronunciation (the default value is 1.0), and the phonetic spelling begins and ends with a word boundary mark (#). The new entries for inflected forms contain TREE and WORDINDEX properties of their own, plus a ROOT-INFLECT property (which indicates the root word of which the new entry is an inflected form), the type of inflection, and the syntactic categories of which the new word is a member. For example,

ROOT-INFLECT (ACCOUNT -S N)

indicates that the entry is the regularly inflected (plural) form of the noun ACCOUNT formed by adding the ending -S. Entries in EXPDICT.FRAGMENT have exactly the same format as in INFLECTED except that the TREE property has been replaced by the property EXPHONES, which records the different phonetic spellings that result from the dictionary expansion. The format for these phonetic spellings is the same as for the TREE property.

The EXPHONES.FRAGMENT file gives an example of the kinds of output that are generated for the phonologist to look at to see the effects of his rules. It consists of two sections. The first gives the original phonetic spelling, the resulting spellings, and the rules used (if any) for each entry in the dictionary. The second section gives for each rule the number of times it applied and a list of the words to which it applied (stored as the value of the atom RULEHISTORY). Finally at the end of the file, a summary of the number of roots, words, and pronunciations is given.

VERDICT.FRAGMENT is an example of the file that is created for the Verification component. The format of an entry in this file is (ndx word pron1 pron2 ... pronn) where ndx is the wordindex of word and each proni is a pronunciation of the word to be considered by the Verification component. Most words have only one such pronunciation, but occasional words have two or more. The format of a pronunciation here consists of a sequence of the elements making up its phonetic spelling followed by a word boundary mark (#).

DICTTEXT.FRAGMENT gives an example of the dictionary file constructed for input to the Lexical Retrieval component. The format here consists of one pronunciation per line, each of which is a sequence:

p <tab> pfeats <tab> pron <tab> word <tab> categories

where p is a pronunciation likelihood, pfeats is a list of pronunciation features (separated by spaces) or NIL if none, pron is the phonetic spelling of the word (with stress marks incorporated into vowels and syllable boundaries removed), and categories is a list of the syntactic categories of which the word is a member (separated by spaces). For subsequent pronunciations of the same word, the common information about the word and its syntactic categories is not repeated.

```
<WOODS>TRAVELDICT.FRAGMENT;1      Tue 26-Oct-76 5:51PM  PAGE 1

[A
       (ART *
        PHONEMES ((OR (EY !2)
                      ((REDUCED)
                       AH !0))) )]
[ABOUT
       (PREP *
        PHONEMES (AX !- * B AW !2 T)
        NUM-ADJ * )]
[ACCOUNT
       (N -S
        PHONEMES (AX !- * K AW !2 N T) )]
[ACL
       (PHONEMES (EY !1 * S IY !1 * EH !2 L)
        SPONSOR * )]
[ACOUSTICAL
       (ADJ *
        PHONEMES (AX !- * K UW !2 * S T IH !0 * K AX !- L) )]

            .
            .
            .

        ----------------------------------------------------

<WOODS>INFLECTED.FRAGMENT;1      Tue 26-Oct-76 3:10PM  PAGE 1

[A
       (ART *
        PHONEMES ((OR (EY !2)
                      ((REDUCED)
                       AH !0)))
        TREE (OR ((1.0)
                  # EY !2 #)
                 ((1.0 REDUCED)
                  # AH !0 #))
        WORDINDEX 3 )]
[ABOUT
       (PREP *
        PHONEMES (AX !- * B AW !2 T)
        NUM-ADJ *
        TREE ((1.0)
              # AX !- * B AW !2 T #)
        WORDINDEX 4 )]
[ACCOUNT
       (N -S
        PHONEMES (AX !- * K AW !2 N T)
        TREE ((1.0)
              # AX !- * K AW !2 N T #)
        WORDINDEX 5 )]
[ACCOUNT-S
       (ROOT-INFLECT (ACCOUNT -S N)
        TREE ((1.0)
              # AX !- * K AW !2 N T -S #)
        WORDINDEX 6 )]
[ACL
       (PHONEMES (EY !1 * S IY !1 * EH !2 L)
        SPONSOR *
        TREE ((1.0)
              # EY !1 * S IY !1 * EH !2 L #)
        WORDINDEX 7 )]
```

```
            .
            .
            .
[A
      (ART *
       PHONEMES ((OR (EY !2)
                     ((REDUCED)
                      AX !0)))
       WORDINDEX 3
       EXPHONES (OR ((1.0)
                     # EY !2 #)
                    ((.7 REDUCED)
                     # AX !- #)
                    ((.5 REDUCED)
                     # AH !0 #)) )]
[ABOUT
      (PREP *
       PHONEMES (AX !- * B AW !2 T)
       NUM-ADJ *
       WORDINDEX 4
       EXPHONES ((1.0)
                 # AX !- * B AW !2 T #) )]
[ACCOUNT
      (N -S
       PHONEMES (AX !- * K AW !2 N T)
       WORDINDEX 5
       EXPHONES ((1.0)
                 # AX !- * K AW !2 N T #) )]
[ACCOUNT-S
      (ROOT-INFLECT (ACCOUNT -S N)
       WORDINDEX 6
       EXPHONES ((1.0)
                 # AX !- * K AW !2 N T S #) )]
[ACL
      (PHONEMES (EY !1 * S IY !1 * EH !2 L)
       SPONSOR *
       WORDINDEX 7
       EXPHONES ((1.0)
                 # EY !1 * S IY !1 * EH !2 L #) )]

            .
            .
            .
```

<WOODS>EXPHONES.FRAGMENT;1 Fri 29-Oct-76 5:41PM PAGE 1

```
            .
            .
            .
[A
      (PHONEMES ((OR (EY !2)
                     ((REDUCED)
                      AH !0)))
       USEDRULES (Z2)
       EXPHONES (OR ((1.0)
                     # EY !2 #)
                    ((.7 REDUCED)
                     # AX !- #)
                    ((.5 REDUCED)
                     # AH !0 #)) )]
[ABOUT
      (PHONEMES (AX !- * B AW !2 T)
       EXPHONES ((1.0)
                 # AX !- * B AW !2 T #) )]
[ACCOUNT
      (PHONEMES (AX !- * K AW !2 N T)
       EXPHONES ((1.0)
                 # AX !- * K AW !2 N T #) )]
[ACCOUNT-S
      (USEDRULES (R-S2)
       EXPHONES ((1.0)
                 # AX !- * K AW !2 N T S #) )]
[ACL
      (PHONEMES (EY !1 * S IY !1 * EH !2 L)
       EXPHONES ((1.0)
                 # EY !1 * S IY !1 * EH !2 L #) )]

            .
            .
            .


[ZERO
      (PHONEMES (Z IY !2 * R OW !0)
       EXPHONES ((1.0)
                 # Z IY !2 * R OW !0 #) )]
)
```

```
  (LISPXPRINT (QUOTE ((V: (RULEHISTORY DICTFILE #ROOTS #WORDS
                                       #PRONUNCIATIONS EXPRATIO))))
              T T)
(LISPXPRINT (QUOTE (V: (RULEHISTORY DICTFILE #ROOTS #WORDS
#PRONUNCIATIONS EXPRATIO))) T)
(DEFINEV
(RULEHISTORY ((FR1 23 COMPUTATIONAL COMPUTE COMPUTE-ED COMPUTE-ING
                   COMPUTE-S CONTINUE-ED CONTINUE-S FUTURE HOUSTON JUNE
                   MASSACHUSETTS MASSACHUSETTS NEW@YORK TUESDAY TUESDAY
                   UNIVAC UNIVAC USUALLY USUALLY USUALLY USUALLY UTAH
                   UTAH)
              (R-ED1.A 2 LAST-ED LIST-ED)
              (R-ED1.B 15 ADD-ED AFFORD-ED ATTEND-ED BUDGET-ED
                       COMPUTE-ED CREATE-ED END-ED ESTIMATE-V-ED
                       NEED-ED OVER-BUDGET-ED PRINT-ED START-ED
                       UNDER-BUDGET-ED VISIT-ED WANT-ED)
              (R-ED2 2 MARK-ED STOP-ED)


            .
            .
            .

              (Z57 9 ANY ANYBODY ANYBODY ANYBODY ANYONE ANYWHERE BONNIE
                   MANY MONEY)
              (Z58 6 FORGOTTEN FORGOTTEN GOTTEN TOTAL TOTAL-ED TOTAL-S)
              (Z5A 3 DO TO YOU)
              (Z6 4 COSELL NOVEMBER OTTAWA PHONOLOGY)
              (Z60 3 ARRANGE ARRANGE CHANGE)
              (Z7 10 AMHERST AMHERST DOLLAR DOLLAR-S FOUR FOURTH HER
                  RECORD RECORD-S RICHARD)
              (Z8A 1 OUR)))
(DICTFILE <SPSYS>TRAVELDICT.;20)
(#ROOTS 491)
(#WORDS 667)
(#PRONUNCIATIONS 1519)
(EXPRATIO 2.277361)
)
STOP
```

```
<WOODS>VERDICT.FRAGMENT;1   Tue 26-Oct-76 3:15PM   PAGE 1

(1 -PAUSE- - # )
(2 S S # Z # AX !- Z # )
(3 A EY !2 # AH !0 # )
(4 ABOUT AX !- * B AW !0 T # )
(5 ACCOUNT AX !- * K AW !2 N T # )
(6 ACCOUNT-S AX !- * K AW !2 N T S # )
(7 ACL EY !1 * S IY !1 * EH !2 L # )
(8 ACOUSTICAL AX !- * K UW !2 * S T IH !0 * K AX !- L # )
(9 ACOUSTICS AX !- * K UW !2 * S T IH !0 K S # )
(10 ACTUAL AE !2 K * CH UW !0 L # AE !2 K * CH UW !0 * AX !- L # )
(11 ACTUALLY AE !2 K * CH UW !0 * L IY !0 # AE !2 K * CH UW !0 * AX !- * L IY !0 # )
(12 ADD AE !2 D # )
(13 ADD-ED AE !2 * D IX !- D # )
(14 ADD-ING AE !2 * D IX !- NX # )
(15 ADD-S AE !2 D Z # )

            .
            .
            .

      ----------------------------------------      ------

<WOODS>DICTTEXT.FRAGMENT;1   Tue 26-Oct-76 3:18PM  PAGE 1

1.0      NIL        -          -PAUSE-   SPECIAL
1.0       UNVOICED             S         -S        AUX SPECIAL V
1.0       VOICED   Z
1.0       STRIFRIC             AX!- Z
1.0      NIL        EY!2       A           ART
.7        REDUCED               AX!-
.5        REDUCED               AH!0
1.0      NIL        AX!- B AW!2 T   ABOUT     NUM-ADJ PREP
.5       NIL        AX!- KA AW!2 T ACCOUNT    N
1.0      NIL        AX!- KA AW!2 N T
.5       NIL        AX!- KA AW!2 TS S         ACCOUNT-S
1.0      NIL        AX!- KA AW!2 N TS S
1.0      NIL        EY!1 S IY!1 EH!2 L        ACL       SPONSOR
.28      NIL        AX!- KA UW!2 S T KA EL!-            ACOUSTICAL
.56      NIL        AX!- KA UW!2 S ST IX!- KA EL!-
.35      NIL        AX!- KA UW!2 S T KA AX!- L
.7       NIL        AX!- KA UW!2 S ST IX!- KA AX!- L
.4       NIL        AX!- KA UW!2 S ST IH!0 KA EL!-
.5       NIL        AX!- KA UW!2 S ST IH!0 KA AX!- L
.7       NIL        AX!- KA UW!2 S ST IX!- K S        ACOUSTICS
.5       NIL        AX!- KA UW!2 S ST IH!0 K S

            .
            .
            .
```

Appendix 3 - Results of Final Run

This appendix lists, for each sentence type and token in the 124 utterance final performance test, the results of running on both BIGDICT-115 (the 1097 words accessible to BIGGRAM) and TRAVELDICT-120 (the 409 words accessible to MIDGRAM) versions of the HWIM system.

The notation N1/N2 in each entry denotes that N1 theories were created and N2 seconds of CPU time were consumed.

Utterances marked "Gave up" were terminated after 150 theories were processed.

What is the budget figure?

RMS102
  BIGDICT: Got: GIVE ME TOO OUR BUDGET FIGURE (1) 34/1943
  TRAVELDICT: Got: GIVE THE BUDGET FIGURE (1)(2) 22/1196
WAW102
  BIGDICT: Correct 23/1745
  TRAVELDICT: Correct 21/1236

(1) IS matched poorly after WHAT.
(2) Semantically identical to the correct theory.

List all trips to California this year.

JJW106B
  BIGDICT: Got: LIST ALL TRIP-S TO PERU FOR HIM THIS YEAR (1) 38/2290
  TRAVELDICT: Correct 15/1137
WAW106
  BIGDICT: Correct 28/2002
  TRAVELDICT: Correct 29/1772

(1) Insufficient stack length in Lexical Retrieval.

What is the registration fee?

RMS110D
  BIGDICT: Got: GIVE THE REGISTRATION FEE (1)(2) 12/1169
  TRAVELDICT: Got: GIVE THE REGISTRATION FEE (1)(2) 12/843
WAW110C
  BIGDICT: Correct 18/1334
  TRAVELDICT: Correct 18/1106

(1) Failed to match IS after WHAT.
(2) Semantically identical to correct theory.

Who's going to IFIP?

RMS119
  BIGDICT: Correct 49/2655
  TRAVELDICT: Correct 42/2022
WAW119B
  BIGDICT: Got: WHO IS GOING TO PISA THEN (1) 38/2365
  TRAVELDICT: Got: WHO IS GOING TO PISA (1) 46/2086

(1) IFIP matches poorly after TO due to unanticipated glottal stop.

When is the next ASA meeting?

RMS125B
  BIGDICT: Gave up (1) 150/4654
  TRAVELDICT: Gave up (1) 150/4337
WAW125C
  BIGDICT: Got: WHEN IS THE NEXT ASSP MEETING (2) 19/1453
  TRAVELDICT: Correct 11/776

(1) Bad match for IS after WHEN and no match for THE due to bad lattice.
(2) Segment lattice had extra segments between ASA and MEETING, thus resulting in a worse score than ASSP MEETING.

How much have we already spent?

RMS126
  BIGDICT: Got: HOW MUCH IS BILL -S EXPENSE (1) 138/5304
  TRAVELDICT: Got: HOW MUCH IS BILL -S EXPENSE (1) 53/2575
WAW126D
  BIGDICT: Correct 24/1587
  TRAVELDICT: Correct 24/1138

(1) Poor match for HAVE followed by a poor verify score.

How much is left?

JJW137
  BIGDICT: Got: WHAT IS LEFT (1)(2) 18/1433
  TRAVELDICT: Got: WHAT IS LEFT (1)(2) 12/933
WAW137D
  BIGDICT: Correct 8/1077
  TRAVELDICT: Correct 8/904

(1) No correct seed words were found.
(2) Semantically identical to the correct theory.

Show me a list of the remaining trips.

RMS178
  BIGDICT: Got: SHOW ALL WALLY -S FIVE REMAINING TRIP-S (1) 79/3210
  TRAVELDICT: Got: SHOW ALL BILL -S FIVE REMAINING TRIP-S (1) 55/2228
WAW178
  BIGDICT: Got: SHOW ANY NEXT OTHER REMAINING TRIP-S (2) 147/5675
  TRAVELDICT: Got: SHOW ME A LIST OF THEIR REMAINING TRIP-S (3) 23/1654

(1) Failed to find ME after SHOW.
(2) No correct seeds. SHOW was found to the left of the seed ANY.
(3) Fails to find THE after OF, although it does find THEIR.

What is the registration fee for the next ACL conference?

RMS180
  BIGDICT: Got: GIVE OUR REGISTRATION FEE FOR THE NEXT ACL CONFERENCE (1) 29/2038
  TRAVELDICT: Got: GIVE OUR REGISTRATION FEE FOR THE NEXT ACL CONFERENCE (1) 27/1412
WAW180
  BIGDICT: Gave up (2) 150/4820
  TRAVELDICT: Got: WHAT IS THE REGISTRATION FOR THE NEXT ACL CONFERENCE (2)(3) 140/3957

(1) GIVE scores better than WHAT, and WHAT never gets seen.
(2) FEE scores better than FOR after REGISTRATION but cannot be followed by FOR due to a 3-to-1 segmentation error.
(3) Semantically identical to the correct theory.

Enter a trip for Bonnie Nash-Webber to Grenoble.

JJW183
  BIGDICT: Correct 19/1709
  TRAVELDICT: Correct 20/1433
WAW183
  BIGDICT: Gave up (1) 150/6058
  TRAVELDICT: Gave up (1) 150/5620

(1) ENTER does not score well.

Please show me John Makhoul's three trips to Pittsburgh.

JJW185
  BIGDICT: Got: SHOW ME ALL OF THOSE THREE TRIP-S TO PITTSBURGH (1)(2) 41/2536
  TRAVELDICT: Got: SHOW ME ALL OF THOSE THREE TRIP-S TO PITTSBURGH (1) 33/1699
WAW185
  BIGDICT: Got: SHOW ME JOHN MAKHOUL -S THREE TRIP-S TO PITTSBURGH (1)(3) 23/1885
  TRAVELDICT: Got: SHOW ME JOHN MAKHOUL -S THREE TRIP-S TO PITTSBURGH (1)(3) 17/1134

(1) PLEASE did not score well and received a poor verify score.
(2) Fails to find JOHN after SHOW ME.
(3) Semantically identical to the correct theory.

When did Craig go to Utah?

JJW270C
  BIGDICT: Correct 35/1914
  TRAVELDICT: Correct 31/1400
WAW270B
  BIGDICT: Correct 57/2720
  TRAVELDICT: Got: WHAT IS THIS PLANE FARE TO OTTAWA (2) 20/1212

What is the plane fare to Ottawa?

JJW272D
  BIGDICT: Correct 15/1434
  TRAVELDICT: Correct 15/1475
WAW272
  BIGDICT: Got: WHAT IS THAT PLANE FARE TO OTTAWA (1) 27/1826
  TRAVELDICT: Got: WHAT IS THAT PLANE FARE TO OTTAWA (1) 13/1264

(1) THAT PLANE scores better than THE PLANE.
(2) Small MAXSEG differences (compared to the BIGDICT run) make THIS look very good. The spanning theory grew from the seed THIS.

Add a new budget item.

RMS275B
  BIGDICT: Correct 15/1091
  TRAVELDICT: Correct 14/943
WAW275
  BIGDICT: Correct 85/3154
  TRAVELDICT: Correct 96/3389

The registration fee is twenty dollars.

RMS277B
  BIGDICT: Got: THEIR REGISTRATION FEE IS 20 DOLLAR-S (1) 33/2074
  TRAVELDICT: Got: THEIR REGISTRATION FEE IS 20 DOLLAR-S (1) 24/1095
WAW277
  BIGDICT: Got: HER REGISTRATION COST FIVE DOLLAR-S (2)(3) 27/1730
  TRAVELDICT: Got: HER REGISTRATION COST FIVE DOLLAR-S (2)(3) 44/1767

(1) THE is a poor seed and is so short that it can't be verified. Also, lattice contains an extra segment before the R of REGISTRATION.
(2) HER scores better than THE.
(3) FEE is a poor match next to REGISTRATION.

List the remaining untaken trips.

JJW278C
  BIGDICT: Gave up (1) 150/4500
  TRAVELDICT: Gave up (1) 150/3684
WAW278
  BIGDICT: Gave up (2) 150/4972
  TRAVELDICT: Correct 51/2152

(1) THE matched poorly to the right of LIST.
(2) Poor matches for both REMAINING and UNTAKEN.

Which trips were canceled?

JJW279D
  BIGDICT: Gave up (1) 150/4972
  TRAVELDICT: Correct 25/1282
WAW279B
  BIGDICT: Got: ERIC -S TRIP-S WERE TEN DOLLAR-S (2) 141/6141
  TRAVELDICT: Got: HIS TRIP-S WERE TEN DOLLAR-S (2) 85/3860

(1) Failed to notice a good match for CANCEL-ED after WERE, due to an inconsistency in the Control/Syntax interface.
(2) WHICH and TRIP-S don't score well and never get seen.

How many trips are there?

RMS280B
  BIGDICT: Correct 43/2114
  TRAVELDICT: Correct 39/1557
WAW280B
  BIGDICT: Correct 9/1000
  TRAVELDICT: Correct 9/622

Cancel Lyn's trip to the ASA meeting.

RMS283
  BIGDICT: Got: GET ONLY ANN -S TRIP TO THE ASA MEETING (1) 71/3176
  TRAVELDICT: Got: GET SUTHERLAND -S TRIP TO THE ASA MEETING (1) 102/3611
WAW283B
  BIGDICT: Correct 56/2851
  TRAVELDICT: Correct 39/1956

(1) No correct seeds, due to po · els in left of lattice.

Give me a list of tl untaken trips.

JJW285
  BIGDICT: Correct 43/2534
  TRAVELDICT: Correct 35/1759
WAW285B
  BIGDICT: Got: GIVE BILL -S FIFTY UNTAKEN TRIP-S (1) 43/2240
  TRAVELDICT: Got: GIVE BILL -S FIFTY UNTAKEN TRIP-S (1) 20/1213

(1) Failed to find ME after GIVE.

Schedule a trip by train to New York.

JJW292D
  BIGDICT: Correct 14/1920
  TRAVELDICT: Correct 14/1600
WAW292B
  BIGDICT: Got: SCHEDULE A TRIP BY TRAIN TO SUNY (1) 39/2889
  TRAVELDICT: Got: SCHEDULE A TRIP BY TRAIN IN MARCH (1) 42/2627

(1) NEW@YORK follows a poor match of TO, while SUNY follows a good one.

What is the one-way air fare from Boston to London?

JJW312
  BIGDICT: Correct 41/2241
  TRAVELDICT: Correct 33/1723
WAW312
  BIGDICT: Gave up (1) 150/5458
  TRAVELDICT: Got: WHAT ARE THESE ONE-WAY AIR FARE-S FROM BOSTON TO LONDON (1) 126/3916

(1) Fails to find IS after WHAT (triple merge in IS).

Create a trip.

RMS315
  BIGDICT: Correct 81/3309
  TRAVELDICT: Correct 53/2056
WAW315
  BIGDICT: Gave up (1) 150/5193
  TRAVELDICT: Got: ARRANGE A TRIP (1)(2) 136/4033

(1) No correct seeds as the lattice was poor around CREATE.
(2) Semantically identical to the correct theory.

Show his trip.

RMS318
  BIGDICT: Got: SHOW LYN -S TRIP (1) 20/1466
  TRAVELDICT: Got: SHOW LYN -S TRIP (1) 19/1112
WAW318
  BIGDICT: Got: SHOW THIS TRIP (2) 35/1999
  TRAVELDICT: Got: SHOW THIS TRIP (2) 30/1514

(1) Failed to match HIS to the right of SHOW.
(2) THIS scores better than HIS to the right of SHOW.

Print her fare.

RMS320
  BIGDICT: Gave up (1) 150/4894
  TRAVELDICT: Gave up (1) 150/3944
WAW320
  BIGDICT: Gave up (2) 150/6359
  TRAVELDICT: Got: FIND THEIR FARE THERE (2) 77/3175

(1) PRINT does not return as a correct seed.
(2) PRINT does not score well due to poor segment lattice.

Show me her trips.

JJW323
  BIGDICT: Correct 14/1142
  TRAVELDICT: Correct 17/921
WAW323
  BIGDICT: Got: SHOW MADER -S TRIP-S (1)(2) 88/3717
  TRAVELDICT: Got: SHOW ME BERT -S TRIP-S (1)(3) 33/1684

(1) Lattice bad for SHOW.
(2) No correct seeds were found. SHOW was found left of MADER.
(3) BERT matches much better than HER.

Was the trip expensive?

JJW330B
  BIGDICT: Got: WAS THAT TRIP EXPENSIVE (1) 34/2140
  TRAVELDICT: Got: WHEN IS THE TRIP TO SDC (2) 138/4107
WAW330
  BIGDICT: Got: WAS THAT TRIP EXPENSIVE (3) 60/3231
  TRAVELDICT: Got: WHAT IS THAT EXPENSE-S (2)(4) 80/3028

(1) Density scoring made WAS THAT look better than WAS THE.
(2) WAS and EXPENSIVE were not allowed to start and end due to an inconsistency in the Control/Syntax interface.
(3) Both WAS and THE score poorly and also get poor verify scores.
(4) Grammar bug accepted this spanning theory.

Do we have a surplus?

RMS333
  BIGDICT: Correct 13/971
  TRAVELDICT: Got: WHO WILL LEAVE THIS MONTH (1) 98/2467
WAW333
  BIGDICT: Control fork broke (2)
  TRAVELDICT: Gave up (1) 150/3605

(1) Failed to notice SURPLUS after DO WE HAVE A due to an inconsistency in the Control/Syntax interface.
(2) Event garbage collector bug.

Why did he visit SDC?

RMS334
  BIGDICT: Got: WHY DID DEYO VISIT SDC (1) 81/3252
  TRAVELDICT: Correct 64/2283
WAW334
  BIGDICT: Gave up (2) 150/5069
  TRAVELDICT: Gave up (2) 150/4631

(1) Failed to find HE after DID due to segmentation error.
(2) Failed to find WHY either as seed or to the left of DID.

Who went to IFIP?

JJW335
  BIGDICT: Correct 16/1187
  TRAVELDICT: Correct 14/769
WAW335
  BIGDICT: Got: WHO WENT TO INDIA (1) 35/2033
  TRAVELDICT: Correct 20/1174

(1) IFIP scored about the same as INDIA (both poor) but was anchored off a worse scoring match of TO.

Are we over the budget?

RMS339
  BIGDICT: Gave up (1) 150/5093
  TRAVELDICT: Got: WERE WE OVER THE BUDGET (2) 64/2778
WAW339
  BIGDICT: Correct 12/1098
  TRAVELDICT: Correct 12/940

(1) No correct seeds were found.
(2) ARE WE scored worse than WERE WE due to a left ending penalty associated with ARE (possible bug).

Show me Bill's trip to Washington.

RMS345
  BIGDICT: Got: SHOW ONLY BELL -S TRIP TO WASHINGTON (1)(2) 98/3607
  TRAVELDICT: Correct 64/2774
WAW345B
  BIGDICT: Got: SHOW ME BELL -S TRIP TO WASHINGTON (2) 42/2069
  TRAVELDICT: Correct 44/1706

(1) ME scores poorly to right of SHOW and to the left of BILL.
(2) BELL scores better than BILL. Verifier ranks them correctly, but the score isn't enough to counteract the Lexical Retrieval scores.

When is the ACL Conference?

JJW346
  BIGDICT: Correct 13/1163
  TRAVELDICT: Correct 13/808
WAW346
  BIGDICT: Correct 23/1636
  TRAVELDICT: Correct 18/1194

Figure his expenses.

RMS347
  BIGDICT: Got: FIGURE THAT EXPENSE-S (1) 26/1893
  TRAVELDICT: Got: FIGURE THAT EXPENSE-S (1) 26/1431
WAW347
  BIGDICT: Got: FIGURE THIS EXPENSE-S (1) 25/1603
  TRAVELDICT: Got: FIGURE THIS EXPENSE-S (1) 21/1206

(1) THIS scores better than HIS. Also, a grammar bug allowed this theory to be accepted; otherwise, would have accepted FIGURE THIS EXPENSE.

Display all taken trips.

RMS348
  BIGDICT: Correct 111/4267
  TRAVELDICT: Correct 58/2228
WAW348
  BIGDICT: Correct 15/1344
  TRAVELDICT: Correct 14/917

I will fly to San Diego.

RMS349
  BIGDICT: Got: WHO WILL FLY TO SAN@DIEGO SOON (1)(2) 33/1805
  TRAVELDICT: Got: WHO WILL FLY TO SAN@DIEGO (1) 38/1694
WAW349
  BIGDICT: Correct 14/1176
  TRAVELDICT: Correct 14/873

(1) WHO scores better than I due to non-optional HH in segment lattice.
(2) Bad segment lattice requires SAN@DIEGO to buy an ending penalty, or else accept another word on the right.

When is Chip's trip?

RMS350
  BIGDICT: Correct 18/1298
  TRAVELDICT: Correct 14/981
WAW350
  BIGDICT: Correct 23/1374
  TRAVELDICT: Correct 19/966

List all fares to Chicago.

JJW351
  BIGDICT: Correct 31/1941
  TRAVELDICT: Correct 29/1541
WAW351
  BIGDICT: Correct 25/1718
  TRAVELDICT: Correct 20/1262

How much did Geoff spend?

JJW352
  BIGDICT: Got: HOW MUCH DID CHIP SPEND (1) 42/2125
  TRAVELDICT: Got: HOW MUCH DID CHIP SPEND (1) 25/1216
WAW352
  BIGDICT: Got: HOW MUCH DID JIM SPEND (1) 20/1546
  TRAVELDICT: Got: HOW MUCH DID JACK SPEND (1) 16/1121

(1) Fails to find GEOFF at all, due to poor labeling in lattice.

List all remaining trips.

RMS353
  BIGDICT: Got: LIST ALL THE EIGHTY TRIP-S (1) 18/1451
  TRAVELDICT: Got: LIST ALL THE NINE TRIP-S (1) 15/1175
WAW353
  BIGDICT: Correct 10/1025
  TRAVELDICT: Correct 10/770

(1) Fails to find REMAINING due to bad segment lattice.

Add a trip to France.

RMS354
  BIGDICT: Correct 20/1412
  TRAVELDICT: Correct 30/1441
WAW354
  BIGDICT: Got: ADD IT TO BERT -S TICKET-S (1) 143/5695
  TRAVELDICT: Got: ADD IT TO THEIR ACTUAL COST-S (1) 62/2634

(1) TRIP and TO scored poorly.

How many trips has Rich taken?

JJW355
  BIGDICT: Gave up (1)(2) 150/4653
  TRAVELDICT: Got: HOW MANY TRIP-S HAS BERT TAKEN (2) 74/2747
WAW355
  BIGDICT: Correct 20/1332
  TRAVELDICT: Correct 18/996

(1) MANY scores poorly causing the searching of many other theories.
(2) BERT matches much better than RICH right of HAS.

Who went to Los Angeles this year?

RMS356
  BIGDICT: Correct 31/2265
  TRAVELDICT: Correct 20/1508
WAW356
  BIGDICT: Correct 14/1345
  TRAVELDICT: Correct 13/996

When is Jack going to San Francisco?

RMS357
  BIGDICT: Correct 20/1643
  TRAVELDICT: Correct 2?/1385
WAW357
  BIGDICT: Got: WHEN IS ELLIOT GOING TO SAN@FRANCISCO (1) 58/2737
  TRAVELDICT: Got: WHEN IS THAT TRIP FOR LYN TO SAN@FRANCISCO (1) 147/5007

(1) Fails to find JACK due to segment lattice problems.

What is the round-trip fare to Chicago?

JJW358
  BIGDICT: Got: WHAT IS GLENN -S AIR FARE TO CHICAGO (1) 54/2644
  TRAVELDICT: Correct 33/1626
WAW358
  BIGDICT: Correct 19/1499
  TRAVELDICT: Correct 27/1463

(1) THE scores poorly after WHAT IS due to poor segment lattice there.

Who went to Santa Barbara in August?

JJW359
  BIGDICT: Correct 52/2200
  TRAVELDICT: Correct 41/1692
WAW359
  BIGDICT: Correct 17/1253
  TRAVELDICT: Correct 12/861

Does the speech budget have a surplus?

JJW360
  BIGDICT: Gave up (1) 150/5231
  TRAVELDICT: Gave up (2) 151/4669
WAW360
  BIGDICT: Gave up (3) 150/5020
  TRAVELDICT: Gave up (3) 150/3752

(1) Gets as far as DOES THE SPEECH BUDGET HAVE A but is swamped with bogus "project names".
(2) Failed to notice SURPLUS after DOES THE SPEECH BUDGET HAVE A due to an inconsistency in the Control/Syntax interface.
(3) Fails to match HAVE after DOES THE SPEECH BUDGET due to poor segment lattice.

Why did Jerry go to Univac?

JJW361
  BIGDICT: Got: WHY DID FEEHRER ATTEND AI@LAB (1) 95/4295
  TRAVELDICT: Gave up (1) 150/5116
WAW361
  BIGDICT: Gave up (1) 150/5502
  TRAVELDICT: Gave up (1) 150/4340

(1) Fails to find JERRY due to missing word boundary rule (D # J => J).

How much did we spend in December?

JJW362
  BIGDICT: Got: ANN WENT TO NICE TILL DECEMBER (1) 61/2847
  TRAVELDICT: Got: WHO WENT TO AUSTIN IN DECEMBER (1) 32/1542
WAW362
  BIGDICT: Got: HOW MUCH DID WE SPEND IN THE SUMMER (2) 55/2596
  TRAVELDICT: Got: HOW MUCH DID WE SPEND IN THE SUMMER (2) 251443

(1) HOW scored poorly and never got to the top of the queue.
(2) DECEMBER scores better than THE but not as good as THE SUMMER. Verifier ranks them correctly, but not enough to offset the other scores.

Schedule a trip for Jack Klovstad to SDC.

JJW365
  BIGDICT: Gave up (1) 150/6037
  TRAVELDICT: Correct 42/2273
WAW365
  BIGDICT: Gave up (2) 150/5936
  TRAVELDICT: Gave up (2) 150/4736

(1) Fails to find KLOVSTAD after JACK.
(2) Fails to find JACK due to segmentation error (3 segments for vowel).

How much money is in the current budget?

JJW366
  BIGDICT: Got: WHAT IS CONNIE -S ENTIRE BUDGET (1) 75/3407
  TRAVELDICT: Gave up (1) 150/4529
WAW366
  BIGDICT: Correct 18/1644
  TRAVELDICT: Correct 15/1105

(1) HOW is found as a seed, but its abysmal score means it never gets seen.

The robot budget has 7 K.

JJW367
  BIGDICT: Gave up (1) 150/5743
  TRAVELDICT: Gave up (1)(2) 150/4300
WAW367
  BIGDICT: Correct 53/2750
  TRAVELDICT: Correct 31/1542

(1) Gets as far as THE ROBOT BUDGET HAS SEVEN but SEVEN buys a very large compatibility penalty (due to a matcher bug).
(2) Fails to match K at the end -- vowel labeled as nasal.

The trip number is 4 3 7 6.

RMS369
  BIGDICT: Got: THE TRIP NUMBER IS 5 3 7 6 (1) 118/5613
  TRAVELDICT: Gave up (2) 150/6063
WAW369
  BIGDICT: Correct 72/3648
  TRAVELDICT: Correct 66/2794

(1) FOUR scores about the same as FIVE but extra fricative before THREE makes FIVE THREE look better than FOUR THREE.
(2) THE scored poorly and was never seen again because of other better-scoring theories.

What trips remain in the speech budget?

JJW370
  BIGDICT: Gave up (1) 150/6138
  TRAVELDICT: Gave up (1) 150/5446
WAW370
  BIGDICT: Correct 14/1436
  TRAVELDICT: Correct 14/976

(1) Failed to find REMAIN after WHAT TRIP-S due to a segmentation problem.

My trip to Mexico cost eight hundred dollars.

JJW371
  BIGDICT: Got: A TRIP TO MEXICO COST A HUNDRED FOUR BUCKS (1)(2)(3) 34/2076
  TRAVELDICT: Gave up (1)(2)(3) 150/4656
WAW371
  BIGDICT: Got: MY TRIP TO MEXICO COST A HUNDRED DOLLAR-S (2) 142/4720
  TRAVELDICT: Got: MY TRIP TO MEXICO COST A HUNDRED DOLLAR-S (2) 47/1886

(1) MY scores very poorly since there was no segment for M.
(2) No T in EIGHT.
(3) DOLLAR matches after HUNDRED but not DOLLAR-S.

Please give me the actual cost of trip number 6 8 7 3.

RMS372
  BIGDICT: Gave up (1) 150/6730
  TRAVELDICT: Syntax fork broke (1)(2)
WAW372
  BIGDICT: Gave up (1) 150/6239
  TRAVELDICT: Got: WHERE IS THE D.C. WORKSHOP ON THE EIGHTH OF NOVEMBER SIXTY SEVEN (1) 119/4614

(1) No correct seeds.
(2) Grammar/dictionary bug.

What is the cost of my trip?

RMS373
BIGDICT: Got: WHY DID DICK GO TO IFIP (1) 70/3202
TRAVELDICT: Got: PLEASE GIVE THEIR COST OF THE TRIP (2) 84/3052
WAW373
BIGDICT: Correct 27/1854
TRAVELDICT: Correct 24/1486

(1) IS scores poorly and WHAT IS never gets to the top of the queue.
(2) IS scores poorly but WHAT IS eventually reaches top of queue, finds THE and is effectively killed by a large compatibility penalty (matcher bug).

How much is in the speech understanding budget?

JJW374
BIGDICT: Gave up (1) 150/6192
TRAVELDICT: Gave up (1) 150/5222
WAW374
BIGDICT: Correct 40/2148
TRAVELDICT: Correct 40/1555

(1) Quickly gets to HOW MUCH IS IN THE SPEECH UNDERSTANDING and fails to find BUDGET, due to poor segmentation and labeling

Where is the next ASA meeting?

JJW375
BIGDICT: Correct 14/1343
TRAVELDICT: Correct 14/869
WAW375
BIGDICT: Correct 19/1593
TRAVELDICT: Correct 15/1078

Lyn's trip to France cost six hundred dollars.

JJW376
BIGDICT: Got: LYN -S TRIP TO FRANCE COSTS SIX HUNDRED DOLLAR-S (1) 43/2654
TRAVELDICT: Got: LYN -S TRIP TO FRANCE COSTS SIX HUNDRED DOLLAR-S (1) 43/2098
WAW376
BIGDICT: Got: GLENN -S TRIP TO FRANCE COSTS SIX HUNDRED DOLLAR-S (1) (2) 31/1971
TRAVELDICT: Got: LYN -S TRIP TO FRANCE COSTS SIX HUNDRED DOLLAR-S (1) 34/1920

(1) The competing events were very close in score although COSTS SIX scored slightly better than COST SIX.
(2) Bad vowel label in LYN scored worse than the initial G in GLENN against a pause.

Show me all the trips to El Paso.

RMS377
BIGDICT: Got: SHOW ALL LEAVITT -S TRIP-S TO EL@PASO (1)(2) 43/2217
TRAVELDICT: Gave up (1) 150/3561
WAW377
BIGDICT: Correct 15/1218
TRAVELDICT: Correct 16/1077

(1) No correct seeds.
(2) SHOW has an L inserted between SH and OW and ALL is matched at the wrong place due to a very deep lattice.

Enter a trip to Grenoble.

RMS378
  BIGDICT: Got: GET THEIR TRIP TO GRENOBLE (1) 59/3024
  TRAVELDICT: Got: GET THEIR TRIP TO GRENOBLE (1) 56/2459
WAW378
  BIGDICT: Correct 11/1120
  TRAVELDICT: Correct 11/848

(1) ENTER scores poorly as a seed and is effectively killed when a poorly scoring A is found on its right.

Appendix 4

Performance Results For Strategy Variations

This appendix presents the results for the strategy variation experiments described in Section F. The set of 10 utterances was selected at random from the larger test set of 124. Note that for these experiments, the "give up" threshold was set to 100 theories rather than the 150 used in the final performance runs.

The notation N1/N2 denotes that N1 theories were created and N2 seconds of CPU time were consumed.

Experiment 1.

Scoring variations using the Left Hybrid strategy with No Verification.

LHQNV: using Quality scoring
LHQDNV: using Quality Density scoring
LHSNV: using Shortfall scoring
LHSDNV: using Shortfall Density scoring

	LHQNV	LHQDNV	LHSNV	LHSDNV
WAW346	CORRECT 33/1790	CORRECT 36/1724	Gave up 100/4130	CORRECT 29/1656
WAW183	Gave up 100/3431	Gave up 100/3785	Gave up 100/4933	Gave up 100/4064
JJW323	CORRECT 18/1147	CORRECT 35/1506	Gave up 100/3886	Correct 30/1511
WAW292B	Incorrect(1) 24/1757	Incorrect(1) 92/4563	Gave up 100/5973	Incorrect(1) 32/2713
WAW351	CORRECT 52/1700	CORRECT 85/2722	Gave up 100/3577	CORRECT 67/2407
WAW119B	Gave up 100/2432	Gave up 100/2685	Gave up 100/4243	Incorrect(2) 92/2845
WAW280D	CORRECT 7/397	Gave up 100/2685	Gave up 100/3686	CORRECT 10/864
WAW348	CORRECT 31/1469	CORRECT 70/2579	Gave up 100/3777	CORRECT 32/1612
WAW370	CORRECT 9/844	CORRECT 42/1519	Gave up 100/4199	CORRECT 15/1393
JJW365	Gave up 100/2564	Gave up 100/3583	Gave up 100/4701	Gave up 100/3341

(1) SCHEDULE A TRIP BY TRAIN TO INDIA

(2) WHO IS GOING TO PISA THEN

Experiment 2.

Scoring variations using the general (middle-out) strategy with No Verification.

QNV: using Quality scoring
QDNV: using Quality Density scoring
SNV: using Shortfall scoring
SDNV: using Shortfall Density scoring

	QNV	QDNV	SNV	SDNV
WAW346	CORRECT 11/549	Gave up 100/3242	Gave up 100/4091	Gave up 100/3255
WAW183	Syntax fork broke (1)	Syntax fork broke (1)	Gave up 100/4431	Gave up 100/3672
JJW323	CORRECT 7/423	CORRECT 54/2146	Gave up 100/3966	CORRECT 23/1263
WAW292B	Incorrect(2) 22/1173	Gave up 100/4410	Gave up 100/4600	Gave up 100/3866
WAW351	Gave up 100/3347	Gave up 100/3541	Gave up 100/3995	CORRECT 79/2777
WAW119B	Gave up 100/4219	Gave up 100/3998	Gave up 100/4810	Gave up 100/3998
WAW280D	CORRECT 7/598	CORRECT 39/1850	Syntax fork broke (3)	CORRECT 32/1502
WAW348	Incorrect(4) 88/2827	Gave up 100/3494	Gave up 100/4041	CORRECT 80/2924
WAW370	CORRECT 9/570	CORRECT 49/2267	Gave up 100/4174	CORRECT 42/1668
JJW365	Gave up 100/2279	Gave up 100/4147	Control fork broke (5)	Gave up 100/3904

(1) Grammar/dictionary bug

(2) SCHEDULE A TRIP BY TRAIN TO INDIA

(3) Grammar bug

(4) DISPLAY TEN TRIP-S

(5) Unknown bug in Control.

Experiment 3.

Effect of heuristics on the general (middle-out) strategy, using Shortfall Density scoring with No Verification.

SD+0: using no additional heuristics
SD+C: using Collision events
SD+G: using Ghosts
SD+GD: using Ghosts and CHOOSEDIR
SD+GDC: using Ghosts, CHOOSEDIR, and Collisions (same as SDNV)

	SD+0	SD+C	SD+G	SD+GD	SD+GDC
WAW346	Gave up 100/3852	Gave up 100/3895	Gave up 100/3864	Gave up 100/3176	Gave up 100/3255
WAW183	Syntax fork broke (1)	Syntax fork broke (1)	Syntax fork broke (1)	Gave up 100/3654	Gave up 100/3672
JJW323	CORRECT 84/3016	CORRECT 29/1753	CORRECT 77/2880	CORRECT 64/2645	CORRECT 23/1263
WAW292B	Syntax fork broke (2)	Syntax fork broke (2)	Gave up 100/4421	Gave up 100/4220	Gave up 100/3866
WAW351	Gave up 100/3611	Gave up 100/3454	Gave up 100/3098	Gave up 100/3456	CORRECT 79/2777
WAW119B	Gave up 100/4322	Gave up 100/4335	Gave up 100/4464	Gave up 100/4044	Gave up 100/3998
WAW280D	CORRECT 39/1862	CORRECT 44/1876	CORRECT 32/1570	CORRECT 31/1790	CORRECT 32/1502
WAW348	Gave up 100/3442	Gave up 100/3312	Gave up 100/3202	CORRECT 79/2669	CORRECT 80/2984
WAW370	CORRECT 39/1853	CORRECT 48/1924	CORRECT 36/1792	CORRECT 35/1709	CORRECT 41/1668
JJW365	Gave up 100/3995	Gave up 100/3900	Gave up 100/3767	Gave up 100/3893	Gave up 100/3904

(1) Grammar/dictionary bug

(2) Grammar bug

Experiment 4.

Effect of Verification strategies, using the Left-Hybrid strategy with Shortfall Density scoring.

LHSDNV: using No Verification
LHSDVD: using Verify-at-Do
LHSDVP: using Verify-at-Pick (the 12 October HWIM strategy)

	LHSDNV	LHSDVD	LHSDVP
WAW346	CORRECT 29/1656	CORRECT 29/1826	CORRECT 23/1636
WAW183	Gave up 100/4064	Gave up 100/4301	Gave up 150/6058
JJW323	CORRECT 30/1511	CORRECT 19/1191	CORRECT 14/1142
WAW292B	Incorrect (1) 32/2713	Incorrect (2) 47/3378	Incorrect (2) 39/2889
WAW351	CORRECT 67/2407	CORRECT 46/1997	CORRECT 25/1718
WAW119B	Incorrect (3) 92/2845	Incorrect (3) 47/2256	Incorrect (3) 38/2365
WAW280D	CORRECT 10/864	CORRECT 9/1064	CORRECT 9/1000
WAW348	CORRECT 32/1612	CORRECT 25/1653	CORRECT 15/1344
WAW370	CORRECT 15/1393	CORRECT 13/1405	CORRECT 14/1436
JJW365	Gave up 100/3341	Gave up 100/3937	Gave up 150/6037

(1) SCHEDULE A TRIP BY TRAIN TO INDIA

(2) SCHEDULE A TRIP BY TRAIN TO SUNY

(3) WHO IS GOING TO PISA THEN

Appendix 5

The following pages contain an alphabetical listing of the base forms in BIGDICT, the larger of the two dictionaries used in HWIM's travel budget management domain. Those items also in TRAVELDICT, the smaller dictionary, are marked with an asterisk.

*-PAUSE-
*-S
*A
A.I.M.
ABORT
*ABOUT
ACAPULCO
*ACCOUNT
ACCURATE
ACCURATELY
*ACL
ACM
*ACOUSTICAL
*ACOUSTICS
*ACTUAL
*ACTUALLY
*ADD
*ADDITIONAL
ADELAIDE
*AFFORD
AFIPS
*AFTER
*AGAIN
AGRICULTURE
*AI
AI@LAB
AIELLO
AIGHES
*AIR
*AIRPLANE
ALABAMA
ALAMOGORDO
ALASKA
ALBANY
ALBUQUERQUE
ALICE
*ALL
ALLEN
*ALLOW
*ALMOST
*ALREADY
*ALSO
*AM
AMARILLO
*AMHERST
*AMOUNT
*AMSDEN
AMSTERDAM
*AN
ANAHEIM
ANALYSIS
ANCHORAGE
*AND
ANDERSON
ANN
ANN@ARBOR
ANOTHER
ANSWER
*ANY
*ANYBODY
*ANYONE
ANYTHING
*ANYWHERE
APPLETON
*APRIL
*ARE
ARGONNE
ARIZONA
ARKANSAS
*ARPA
*ARRANGE
ARTHUR
*ASA
ASIS
ASPEN
*ASSOCIATION
ASSP
*ASSUME
ATHENS
ATLANTA
ATLANTIC@CITY
*ATTEND
AUCKLAND
*AUGUST
AUGUSTA
AURIGEMMA
*AUSTIN
AUSTRALIA
AUSTRIA
*AUTO
*AVAILABLE
*AVERAGE
*AWAY
*BAGHDAD
BALTIMORE
BANGKOK
BARBARA
BARON
BARRY
*BATES
*BBN
*BE
BECKER
*BEEN
*BEFORE
*BEGIN
*BEIRUT
BELGIUM
BELL
BELL@LABS
BERKELEY
BERLIN
BERLINER
BERMUDA
*BERT
*BETWEEN
*BILL
BILLY
BINGHAMTON
BIOPSY
BLADDER
BLOOMINGTON
BOB
BOBROW
BOCA@RATON
BOISE
BOLOGNA
BONN
BONNEVILLE
*BONNIE
BOOK
*BOSTON
*BOTH
BOULDER
BOWLING@GREEN
BRACHMAN
BRAZIL
BRAZILIA
*BREAKDOWN
BRENDA
BRIGHAM@YOUNG
BROWN
*BRUCE
BUCHAREST
BUCKS
BUDAPEST
*BUDGET
*BUDGETED
BUDIANSKY
BUFFALO

BURCHFIEL
BURLINGTON
BURTON
*BUS
*BY
CAIRO
CALCULATE
CALCUTTA
*CALENDAR
CALGARY
*CALIFORNIA
CALLEVA
CALTECH
CALVIN
CAMBRIDGE
*CAN
*CANADA
*CANCEL
CAPE@TOWN
*CAR
CARBONELL
CARL
*CARNEGIE
*CARNEGIE-MELLON
CAROL
CATALINA
CATHY
*CENT
CENTER
CHAMPAIGN
*CHANGE
*CHARGE
CHARLENE
CHARLES
CHARLOTTE
CHATTANOOGA
CHERRYHILL
CHEST
CHEYENNE
*CHICAGO
CHILE
*CHIP
CHIPMAN
CHRISTCHURCH
CINCINATTI
*CITY
CLARENCE
CLEARWATER
CLEMENTS
CLEVELAND
CLINIC
*CMU
*COAST
CODE
*COLARUSSO
COLLEGE@PARK
COLLINS
COLORADO
COLORADO@SPRINGS
COLUMBUS
COMBS
COMMERCE
COMPCON
*COMPRESSION
*COMPUTATIONAL
*COMPUTE
COMPUTER
*CONFERENCE
CONNECTICUT
CONNIE
CONSULTING
*CONTAIN
*CONTINUE
*CONTRACT
*COOK
COPENHHAGEN
CORNELL
CORPUS@CHRISTI
CORRECT
CORRELATE
*COSELL
*COST
*COST-PAST
*COSTS
COTCO
*COULD
*COUNTRY
*CRAIG
*CREATE
*CRETE
CUNY
*CURRENT
CZECHOSLOVAKIA
*D.C.
DALLAS
DAN
DARGAN
DARTMOUTH
DARYLE
DAVE
DAVID
*DAY
DAYTON
DE@KLEER
DE@PESA
DECATUR
*DECEMBER
DECISION-MAKING
*DEFICIT
DELAWARE
DELETE
DELHI
DENMARK
*DENNIS
DENVER
DEPART
DETROIT
DEVELOPMENT
DEYO
DICK
*DID
DIFFERENT
DISCRIMINATION
DISKIN
*DISPLAY
DISREGARD
DIVIDE
*DO
DODDS
*DOES
*DOLLAR
*DOMESTIC
DON
*DON-T
*DONE
DOUG
DREXEL
DUBROVNIK
DUBUQUE
DULUTH
DUNCAN
DUNDEE
*DURING
*EACH
*EARLY
*EAST
EAST@LANSING
EDINBURGH
EDMONTON
EGYPT

*EIGHT
*EIGHTEEN
*EIGHTEENTH
*EIGHTH
*EIGHTY
*EITHER
*EL@PASO
*ELEVEN
*ELEVENTH
ELLIOTT
ELSIE
EMOTIONAL
*END
*ENGLAND
*ENOUGH
*ENTER
ENTIRE
ERIC
ERIE
*ESTIMATE-N
*ESTIMATE-V
EUGENE
EUROPE
*EUROPEAN
*EVERY
EVERYBODY
*EVERYONE
*EVERYTHING
EXACT
*EXACTLY
*EXPENSE
*EXPENSIVE
FACTOR
FAIRBANKS
*FALL
*FARE
*FEBRUARY
*FEE
FEEHRER
FELDMAN
FEURZEIG
*FIFTEEN
*FIFTEENTH
*FIFTH
*FIFTY
*FIGURE
*FINAL
*FIND
*FINDS
*FIRST
*FISCAL
*FIVE
FLAGSTAFF
*FLEW
*FLIES
FLINT
FLORENCE
*FLORIDA
*FLOWN
*FLY
*FOR
*FOREIGN
*FORGET
*FORGETS
*FORGOT
*FORGOTTEN
FORT@LAUDERDALE
FORT@WAYNE
FORT@WORTH
FORTMANN
*FORTY
*FOUND
*FOUR
*FOURTEEN
*FOURTEENTH
*FOURTH
*FRANCE
FRANK
FRANKFORT
FREEMAN
*FRIDAY
*FROM
*FUTURE
GAIL
GAINESVILLE
GALLUP
GALVESTON
*GAVE
GENEVA
*GEOFF
*GEOFFREY
GEORGE
GEORGIA
GERMANY
*GET
*GETS
*GIVE
*GIVEN
*GIVES
GLENN
*GO
*GOES
*GONE
*GOT
*GOTTEN
*GOULD
GRAESSER
GRAND@RAPIDS
GREEN
GREEN@BAY
GREENVILLE
GREG
*GRENOBLE
GRIGNETTI
*GROUP
*HAD
HAIFA
HARRIS
HART
HARTLEY
HARVARD
*HAS
HASKINS
HAUSMANN
*HAVE
HAWAII
HAWKINS
*HE
HEALTH
HEARING
HEART
HEDTLER
HENDERSON
*HER
*HERE
*HIM
*HIS
HOLLAND
HOLMDEL
HONG@KONG
HONOLULU
HOT@SPRINGS
*HOUSTON
*HOW
HRA
HUGGINS
HUMAN
HUMREL
*HUNDRED
HUNGARY

*HYPOTHETICAL
*I
IBM
ICAI
*ICCL
IDAHO
IDAHO@FALLS
*IEEE
*IFIP
*IJCAI
ILLINOIS
*IN
INDIA
INDIANA
INDIANAPOLIS
INEXPENSIVE
INITIAL
*INTERNATIONAL
IOWA
*IRAQ
*IS
IST
ISRAEL
*IT
*ITALY
*ITEM
ITEMIZE
ITHACA
*JACK
JACKSON
JACKSONVILLE
JAIME
JAMAICA
JAMES
*JANUARY
JAPAN
*JERRY
JERUSALEM
JIM
JOAN
JOE
JOHAN
JOHANNESBURG
*JOHN
JOHNS@HOPKINS
JOHNSON
JONES
JPL
*JUAREZ
JUDY
JULIE
*JULY
*JUNE
JUNEAU
*JUST
*K
KALAMAZOO
KALIKOW
KANSAS
KANSAS@CITY
KEEP
KEEPS
KEN
KENTUCKY
KENYA
KEPT
KEY@WEST
KHUEN
KIDSTON
KINGSTON
*KLATT
*KLOVSTAD
*KNEW
*KNOW
*KNOWN
*KNOWS
KNOXVILLE
KONA
KYOTO
*L.A.
LA@PAZ
LAFAYETTE
LANCASTER
LARAMIE
LAS@VEGAS
*LAST
*LATE
*LAURA
LAUSANNE
LAWRENCE
*LEAVE
*LEAVES
LEAVITT
*LEBANON
LEEDS
*LEFT
LENINGRAD
*LESS
LEVISON
LEWIS
LIEGE
LIMA
LINCOLN
LINCOLN@LABS
*LINDA
*LINGUISTICS
LISBON
LISP
*LIST
LITTLE@ROCK
LOGO
LOIS
*LONDON
*LONG
LOS@ALAMOS
*LOS@ANGELES
LOUISIANA
LOUISVILLE
LOVETT
LUKAS
*LYN
*MADE
MADELEINE
MADER
MADISON
MADRID
MAINE
*MAKE
*MAKES
*MAKING
MALAYSIA
MAN-MACHINE
MANCHESTER
MANILA
*MANY
*MARCH
MARIO
*MARK
MARSEILLE
MARTIN
MARY
MARY@ANN
MARYLAND
*MASSACHUSETTS
MATTHEW
MAURER
*MAY
*ME
MEDFORD
*MEETING

MELBOURNE
MEMPHIS
MENLO@PARK
MERRIAM
*MEXICO
MEXICO@CITY
*MIAMI
MIAMI@BEACH
MICHAEL
MICHIGAN
MIKE
MILAN
*MILEAGE
MILLER
MILWAUKEE
MINNEAPOLIS
MINNESOTA
MINUS
*MISCELLANEOUS
MISSISSIPPI
MISSOULA
*MISSOURI
MIT
MITNICK
MNEMO
MODIFY
*MONDAY
*MONEY
MONTANA
MONTE@CARLO
*MONTEREY
MONTGOMERY
*MONTH
MONTREAL
*MORE
MORSE
MOSCOW
*MUCH
MULLARKEY
MULTIPLY
MURPHY
MURRAY@HILL
MUSIC
*MUST
*MY
MYER
NAIROBI
NASA
*NASH-WEBBER
NASHVILLE
NATO
NBS
NCC
NCI
*NEARLY
NEBRASKA
*NEED
NELLEKE
NEVADA
*NEW
NEW@BRUNSWICK
NEW@DELHI
NEW@HAMPSHIRE
NEW@HAVEN
NEW@JERSEY
NEW@MEXICO
NEW@ORLEANS
*NEW@YORK
NEW@YORK@CITY
NEW@ZEALAND
NEWPORT@NEWS
*NEXT
NICE
*NICKERSON
NINCDS
NINDS
*NINE
*NINETEEN
*NINETEENTH
*NINETY
*NINTH
NLS
*NO
NOME
*NONE
NORFOLK
NORMAL
*NORMALLY
NORTH@CAROLINA
NORTH@DAKOTA
NORWAY
*NOT
NOTRE@DAME
*NOVEMBER
*NOW
*NUMBER
*OCTOBER
*OF
OGDEN
*OH
OHIO
*OK
OKLAHOMA
OKLAHOMA@CITY
OLIVER
OLYMPIA
OMAHA
*ON
*ONE
*ONE-WAY
*ONLY
ONR
*OR
ORDINARILY
OREGON
ORLANDO
ORLY
OSLO
*OTHER
*OTTAWA
*OUR
*OUT
*OUTSTANDING
*OVER
*OVER-BUDGET
*OVERHEAD
OXFORD
PACKET
PADUCAH
PAIGE
PALM@BEACH
PALM@SPRINGS
*PALO@ALTO
*PARIS
PASADENA
*PAST
PATRICK
PAUL
PEARL
PENDLETON
*PENNSYLVANIA
PENSACOLA
*PEOPLE
PEORIA
*PERDIEM
PERKINS
*PERSON
PERTH
PERU
PEW

*PHILADELPHIA
PHILIPPINES
PHOENIX
*PHONOLOGICAL
*PHONOLOGY
PICKETT
PIERRE
*PISA
*PITTSBURGH
*PLAN
*PLANE
*PLEASE
PLUMMER
*PLUS
POCATELLO
POLAND
PORTLAND
PORTUGAL
POUGHKEEPSIE
PRAGUE
PRECEDING
PRECISE
PRECISELY
PRESCOTT
PRESENTLY
PREVIOUS
PREVIOUSLY
PREWETT
PRICE
PRINCETON
*PRINT
*PROJECT-N
*PROPOSED
PROVIDENCE
PROVO
PUERTO@RICO
PULLMAN
PURDUE
*PURPOSE
*PUT
*PUT-PAST
*PUTS
*QUARTER
QUEBEC
QUEBEC@CITY
*QUIT
RADIO
RALEIGH
RAND
RATIO
*RAY
*RECENT
*RECENTLY
*RECOGNITION
*RECORD
REGINA
*REGISTRATION
REGULARLY
*REMAIN
*REMAINDER
*REMAINING
RENO
REPORT
*REST
RETURN
RHODE@ISLAND
*RICH
*RICHARD
RICHMOND
RIO
RIO@DE@JANEIRO
ROANOKE
ROBERT
*ROBOT
ROCHESTER
ROLLA
ROLLINS
ROME
RON
RONALD
ROSS
*ROUND-TRIP
ROURKE
RUMANIA
RUSSELL
*RUSSIA
RUSTY
RUTGERS
SACRAMENTO
SAGAMORE
SALEM
*SALT@LAKE
SALT@LAKE@CITY
SAN@ANTONIO
*SAN@DIEGO
*SAN@FRANCISCO
SAN@JOSE
SAN@JUAN
*SANTA@BARBARA
SANTA@CRUZ
SANTA@FE
SANTA@MONICA
SANTIAGO
SAO@PAULO
SARAJIAN
SASKATOON
*SATURDAY
SAVANNAH
SCANDORA
SCHANTZ
*SCHEDULE
SCHIFF
SCHOLAR
*SCHWARTZ
SCOTLAND
SCREENING
SCRL
*SDC
SEATTLE
SEAWAY
*SECOND
SELECTION
SELFRIDGE
*SEND
SEPARATE-ADJ
*SEPTEMBER
*SEVEN
*SEVENTEEN
*SEVENTEENTH
*SEVENTH
*SEVENTY
SEVERAL
*SHE
SHEILA
SHELLY
SHORT
*SHOULD
*SHOW
*SHOWN
SHREVEPORT
SILVER@SPRING
*SINCE
SINGAPORE
SIOUX@CITY
*SIX
*SIXTEEN
*SIXTEENTH
*SIXTH
*SIXTY
*SOCIETY

*SOME
*SOMEBODY
*SOMEONE
SOMETHING
*SOMETIME
SOON
SOPHIE
SOUND
SOUTH@AFRICA
SOUTH@CAROLINA
SOUTH@DAKOTA
SPAIN
SPEAR
*SPEECH
*SPEND
*SPENDS
*SPENT
*SPLIT
*SPLIT-PAST
*SPLITS
SPODICK
SPOKANE
*SPRING
SPRINGFIELD
SRI
ST@JOHN
*ST@LOUIS
ST@PAUL
ST@PETERSBURG
ST@THOMAS
STANFORD
*START
*STATE
*STATUS
*STEERING
STEVE
STEVENS
STILLWATER
*STOCKHOLM
STOCKTON
STOCKWELL
*STOP
STRESS
STROLLO
STUDY
SUBTRACT
SUE
*SUFFICIENT
SUM
*SUMMER
SUN@VALLEY
*SUNDAY
SUNY
*SUPPOSE
*SUPPOSITION
*SUR
*SURPLUS
SURVEY
SUSAN
SUSSMAN
*SUTHERLAND
SWARTHMORE
*SWEDEN
SWITZERLAND
SYDNEY
SYMPOSIUM
SYRACUSE
SYSTEM
TACOMA
TAIPEI
TAIWAN
*TAKE
*TAKEN
*TAKES
TALLAHASSEE
TAMPA
TAOS
TED
TEL@AVIV
*TELL
*TEN
TENEX
TENNESSEE
*TENTH
TEST
TESTING
*TEXAS
TEXAS@CHRISTIAN
THAILAND
*THAN
*THANK@YOU
*THAT
*THE
*THEIR
*THEM
*THEN
*THERE
*THESE
*THEY
*THIRD
*THIRTEEN
*THIRTEENTH
*THIRTIETH
*THIRTY
*THIS
THOMAS
THORNTON
*THOSE
*THOUSAND
*THREE
*THROUGH
*THURSDAY
TICKET
TIERNAN
TILL
TIMES
*TO
TODAY
TOKYO
TOLEDO
TOM
TOMLINSON
TOMORROW
TONY
*TOO
*TOOK
TOPEKA
TORONTO
*TOTAL
TOULOUSE
*TRAIN
TRANSCRIBER
*TRANSCRIPTION
*TRAVEL
TRENTON
*TRIP
TROY
TUCSON
*TUESDAY
TULSA
*TURIN
*TWELFTH
*TWELVE
*TWENTIETH
*TWENTY
TWIN@FALLS
*TWO
TYPICAL
TYPICALLY
UCLA

ULMER
*UNANTICIPATED
*UNBUDGETTED
*UNDER
*UNDER-BUDGET
*UNDERSTANDING
*UNEXPECTED
*UNIVAC
*UNTAKEN
UNTIL
*UPCOMING
*US
USC
USSR
*USUALLY
*UTAH
UTICA
VANCOUVER
VERACRUZ
VERIFY
VERMONT
VICTOR
VICTORIA
VIENNA
VIRGIN@ISLANDS
VIRGINIA
VIRGINIA@BEACH
*VISIT
VISUAL
VOCODER
VOICE
WALLA@WALLA
WALLY
WALTER
*WANT
WARSAW
*WAS
*WASHINGTON
WATERLOO
*WAY
*WE
*WEDNESDAY
*WEEK
WELLINGTON
*WENT
*WERE
*WEST
WEST@LAFAYETTE
WEST@PALM@BEACH
WEST@VIRGINIA
*WHAT
*WHEN
*WHERE
*WHICH
*WHITE@PLAINS
*WHO
WHOLE
*WHOM
*WHOSE
*WHY
*WILL
WILLIAMS
WILLIAMSBURG
WILMINGTON
WINCHESTER
WINNIPEG
WINSTON@SALEM
*WINTER
*WISCONSIN
*WITH
*WOLF
*WOODS
WORK
*WORKSHOP
*WOULD
WYOMING
X-RAY
YALE
*YEAR
*YES
YESTERDAY
YORKTOWN
YORKTOWN@HEIGHTS
*YOU
*YOUR
YUGOSLAVIA
*ZERO
ZUE

Appendix 6

Dictionary Expansion - A User's Guide

In this section, we give the sequence of terminal interactions involved in producing the various dictionary files required for running the system. The reader without a specific technical need for reading this section is invited to skip it.

Dictionary expansion generally begins by calling a subsystem called RULETESTER.SAV, which currently resides on directory [BBND]<CWOODS>. This subsystem is a LISP sysout that contains a set of phonological rules already loaded with the basic Bobrow-Fraser rule tester and the BBN extensions.*

There are two versions of the basic rule expansion function: EXPLIST operates entirely in-core and is useful for relatively small dictionaries. We have run dictionaries up to 500 words with it, but there is very little extra room for working storage. EXPFILE, the second version, operates incrementally from a file, sending its output to another file. The description of the former in [Woods, 1975c] is still accurate and will not be repeated here. The latter has changed somewhat since then, as will be described below. The advantage of the in-core expansion is that the entire dictionary is left in core after the expansion, allowing a phonologist to explore anything that he finds strange in the output summary by tracing a re-expansion using the EXPWORD function. In the incremental version, he must first use a function GETWORD to load into core any word that he wishes to re-expand before using EXPWORD.

- - - - - - - - - - - - - - - -

*A basic loadup of LISP plus FLIP (a pattern-matching language embedded in LISP) plus the Bobrow-Fraser rule tester is saved in the sysout file BASICRULETESTER.SAV;1, which currently resides in directory [BBND]<WOODS>. This sysout is built on an ancient version of INTERLISP, namely, LISP.SAV;24. Bringing the rule tester up on a newer version of INTERLISP would require a new version of FLIP, which would have to be provided by Warren Teitelman at Xerox PARC. The symbolics of the Bobrow-Fraser rule tester, on a file [BBND]<WOODS>PHON would also probably have to be modified to run on a current LISP. Symbolics for this file may also live at Xerox PARC and at Speech Communications Research Laboratory in Santa Barbara. The BBN extensions to the rule tester live on the file PRULES, which again might require some modifications to run on a newer version of LISP, but I believe the changes there would not be too difficult. Since the dictionary expansions are only run occasionally and our resources were short, we have not made any attempt to bring up a newer version of the rule expansion system. --W. Woods.

EXPFILE, which invokes incremental dictionary expansion from a file, takes six arguments -- FILE, RULETREE, TFL, TREEPROP, VERFLAG, and JACKFLAG. The first is the name of the file to be expanded, the second is the name of a list of rules to be used for the expansion, and the third is a traceflag to be set to T when a trace of the individual rule match attempts is desired for rule debugging and testing. TREEPROP is an indicator of the property on which the pronunciation to be expanded is stored and indicates which phase of the dictionary expansion is being requested. (The dictionary files are stored in a symbolic analog of the LISP property list, consisting of property-name/property-value pairs for each word.) VERFLAG indicates whether a VERDICT file for the verification component is desired, and JACKFLAG indicates whether a DICTTEXT file for the lexical retrieval component is desired. If the expansion is being done for testing of the rules only and not to produce input dictionary files for the speech understanding system, these last two arguments would be NIL.

All of the dictionary expansion routines make a certain assumption about the format of the name of the dictionary file to be expanded and the names of the files that they produce. Specifically, assuming that the dictionary to be expanded has the ordinary TENEX file name convention FILE.EXT;n (where FILE is the basic file name, EXT is a file extension added to the name, and n is version number of the file), the rule expansion system names all of its files with an extension of FILE-EXT!n, which it constructs from the input file name, extension, and version number. For example, if the dictionary file being expanded was named TRAVELDICT.;19 (no extension), then the expanded dictionary would be named EXPDICT.TRAVELDICT-!19. We will use this particular file name throughout the examples in this section.

In our current set of phonological rules, there are three lists of rules: INFLECTLIST, ZUELIST, and APRLIST, for the three successive phases of expansion. The first phase is performed by typing the command:

```
EXPFILE(TRAVELDICT.;19 INFLECTLIST NIL NIL T)
```

(All of the arguments to EXPFILE beyond the last one typed will be defaulted to NIL). This will produce the following files:

INFLECTED.TRAVELDICT-!19.;1
EXPDICT.TRAVELDICT-!19.;1
EXPHONES.TRAVELDICT-!19.;1
VERDICT.TRAVELDICT-!19.;1

(assuming older versions of the same file names did not already exist). The last is the dictionary file that is used by the Verification component. The first is an intermediate file that results from adding the inflected words to the original root words and sorting. It can be used for later expansions without having to go through this process again. (For example, if we were to edit some of the rules on INFLECTLIST and want to do the expansion again, we could type:

EXPFILE(INFLECTED.TRAVELDICT-!19 INFLECTLIST NIL TREE T)

where the argument TREE indicates that this is an expansion of a previously expanded file and that the property on which the pronunciations are stored in the file is TREE.) The EXPDICT file produced is in the appropriate form for the Syntactic and Control components of the system to use. The EXPHONES file contains a summary of the rule applications for use by the phonologist.

The second phase of expansion, the main phonological rule expansion on the output of the first phase, is initiated by typing:

EXPFILE(EXPDICT.TRAVELDICT-!19 ZUELIST NIL EXPHONES)

This indicates that the pronunciations to be expanded are those that were placed on the EXPHONES property by the previous expansion and that the list of rules to use this time is ZUELIST. As a result of this expansion, the following files would be produced:

APRDICT.TRAVELDICT-!19.;1
APRPHONES.TRAVELDICT-!19.;1

These files are exactly like the EXPDICT and EXPHONES files that were produced by the previous expansion, except they they include the effects of the additional phonological rules in ZUELIST. The APRPHONES file produced at this point is for the phonologist to look at for debugging rules, and the APRDICT file is for input to the next phase.

The final APR expansion is initiated by typing:

```
EXPFILE(APRDICT.TRAVELDICT-!19 APRLIST NIL EXPHONES NIL T)
```

where the final argument indicates that a DICTTEST file for the Lexical Retrieval component is to be made. This expansion will produce the files:

```
APRDICT.TRAVELDICT-!19.;2
APRPHONES.TRAVELDICT-!19.;2
DICTTEXT.TRAVELDICT-!19.;1
```

where DICTTEXT is the file to be used by the Lexical Retrieval component, and the APRDICT and APRPHONES files are as before, except that they contain the results of the APR-rule expansion.

Official Distribution List

Contract N00014-75-C-0533

Defense Documentation Center
Cameron Station
Alexandria, Virginia 22314

Office of Naval Research
Information Systems Program
Code 437
Arlington, Virginia 22217

Office of Naval Research
Code 1021P
Arlington, Virginia 22217

Office of Naval Research
Branch Office, Boston
495 Summer Street
Boston, Massachusetts 02210

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, Illinois 60605

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, California 91106

New York Area Office
715 Broadway - 5th Floor
New York, New York 10003

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D. C. 20375

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Office of Naval Research
Code 455
Arlington, Virginia 22217

Office of Naval Research
Code 458
Arlington, Virginia 22217

Naval Electronics Lab. Center
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation and Mathematics Dept.
Bethesda, Maryland 20084

Captain Grace M. Hopper
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson
Technical Director
Information Systems Division (OP-91T)
Office of Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Projects Agency
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Commanding Officer
Naval Air Development Center
Warminster, Pennsylvania 18974

Professor Omar Wing
Dept. of Electrical Engineering
Columbia University
New York, New York 10027

Assistant Chief for Technology
Office of Naval Research
Code 200
Arlington, Virginia 22217